Concepção e Simulação de uma Célula Robotizada para Acabamentos de Solas de Calçado

Nuno José Eleutério da Silva Moita

Relatório do Projecto Final

Orientadores:

Prof. António Mendes Lopes
Prof. Paulo Augusto Ferreira de Abreu

Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Mecânica

Opção de Automação

Julho de 2009

À minha "Abuelita"

Resumo

Hoje em dia, o mercado português disponibiliza inúmeros tipos de calçado, oferecidos essencialmente por empresas estrangeiras. Este facto constitui um problema para a indústria portuguesa do calçado, dado a sua incapacidade no acompanhamento da diversidade oferecida pela concorrência estrangeira. Os motivos desse facto são, essencialmente, a elevada dependência da Indústria Nacional no uso de processos manuais, exigindo uma elevada qualificação de mão-de-obra. Desta dependência são frutos as quebras de produtividade, causa da baixa flexibilidade apresentada pelos operadores.

Nesta investigação é efectuado o estudo de uma solução robotizada que assenta no uso da tecnologia de controlo de força para realizar o acabamento de solas duplas de sapatos, como forma de contrariar a dependência da mão-de-obra qualificada na indústria do calçado.

Em primeira instância, foi efectuada uma contextualização acerca da situação geral da robótica na indústria do calçado, onde se expõem as soluções comerciais e não comerciais encontradas.

Seguidamente, é apresentado o estudo de possíveis configurações de células robotizadas, que permitam a implementação do processo de acabamento de solas na indústria do calçado.

Finda esta primeira abordagem, foi efectuada uma breve contextualização acerca dos diferentes tipos de programação de robôs, seguindo-se a apresentação do princípio de funcionamento do *software* de concepção e simulação interactiva utilizado: o *RobotStudio*.

Após levantamento e estudo dos procedimentos necessários, procedeu-se à modelação de uma célula robotizada em ambiente virtual (no *RobotStudio*). Esta modelação foi baseada na configuração da célula robotizada existente no laboratório de robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto. Desta forma, foi possível realizar a simulação do processo de acabamento de solas duplas de sapatos.

Para além da simulação do processo de acabamento, idealizou-se, concebeu-se e implementou-se uma solução gráfica para a consola do robô que permite a execução do processo de acabamento de sapatos por pessoal não qualificado.

Para finalizar, foi tomada a iniciativa de se efectuarem testes laboratoriais de modo a obter-se uma maior sensibilidade na implementação do processo em estudo.

Conception and simulation of a Robotic Cell for footwear's sole finishing

Abstract

Nowadays, the market offers innumerable shoe's types, essentially offered by foreign companies. This fact constitutes a problem for the Portuguese footwear's industry due to its incapacity to keep up this diversity. The main motives of this cause are essentially the huge dependency on handmade processes, requiring highly skilled workmanship. This dependency leads to productivity breaks, cause of lower flexibility showed by the operators, when, for example, a new process is added to the production line.

This investigation has born as a form to contradict this dependency, demonstrating a robotic solution based on force control's technology to realize footwear's sole finishing with two soles.

The proposed solution includes several layouts' studies of robotics cells, as well as an implementation on a virtual environment (on the software RobotStudio) of a robotic cell similar to the existing in the Mechanical Department of Engineering Faculty of Porto's University in order to demonstrate the proposed solution.

Verified the process' simulation, it has been developed an application for the robot's console to allow non skilled operators to proceed to the footwear's sole finishing.

To finalize, it has been done laboratorial tests on a real robot in order to obtain more sensibility on the practicable application of this case study.

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus orientadores, Professor Doutor António Mendes Lopes e Professor Doutor Paulo Abreu, a disponibilidade ao longo deste semestre, demonstrando uma incansável dedicação em todas as fases deste trabalho.

Agradeço ao coordenador da opção de Automação, o Professor Doutor Francisco Freitas, pelo acompanhamento feito ao longo deste período.

À empresa Armipex, pela amabilidade de nos ter cedido sapatos para a realização de testes finais.

Quero, ainda, agradecer ao Sr. Ramalho pela sua disponibilidade na solução de pequenos problemas mecânicos surgidos.

Ao núcleo de trabalho deste semestre constituído pelo Tiago Ramos e pelo Tiago Teixeira, agradeço pelos bons momentos e companheirismo demonstrado.

Agradeço aos meus amigos de Erasmus (Kaunas, Lituânia), por me terem proporcionado os melhores momentos enquanto estudante da Faculdade de Engenharia da Universidade do Porto.

Ao meu avô Herlander Silva, agradeço a constante partilha de conhecimentos, sem nunca ter negado uma mão amiga nos momentos mais difíceis.

Um especial agradecimento à minha recentemente falecida avó Carmen Silva, por tudo o que fez em vida para me ensinar a ser uma pessoa melhor todos os dias, e por recentemente me ter ensinado o que é o Amor Eterno.

A todos os meus amigos, em especial à Ana Bastos e ao Filipe Batista, um obrigado.

Por último, gostaria de agradecer aos meus pais, José Moita e Carmo Silva, e à minha irmã, Marília Moita, visto que sem eles, seria impossível estar a escrever estas palavras. Desejo transmitir ao meu pai, com este agradecimento, que com força de vontade se conseguem realizar todos os sonhos.

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

1 Introdução Geral e Objectivos

A indústria portuguesa do calçado sofreu, nos últimos anos, mudanças importantes na sua estrutura. O forte crescimento registado nas últimas três décadas ocorreu devido aos baixos custos de produção que Portugal apresentava, permitindo o investimento por parte de empresas estrangeiras. No entanto, estas empresas, que outrora investiram fortemente na indústria portuguesa, não optaram por participar numa nova estratégia quando os custos de produção aumentaram. Deste modo, assistiu-se a uma deslocalização da sua produção para países onde pudessem continuar a praticar os mesmos custos de produção que os trouxeram para cá. As empresas nacionais, nomeadamente as de pequena e média dimensão, devido à pressão competitiva acrescida e incapacidade de conseguirem acompanhar os novos desafios do mercado, acabaram por não resistir. Destes factores resultou uma queda na casa dos vinte, a vinte e cinco porcento, nas actividades do sector do calçado afectando o número de empresas, o emprego, a produção, e a exportação, tal como se pode observar na Figura 1.1 e Figura 1.2 (CEGEA, 2007).
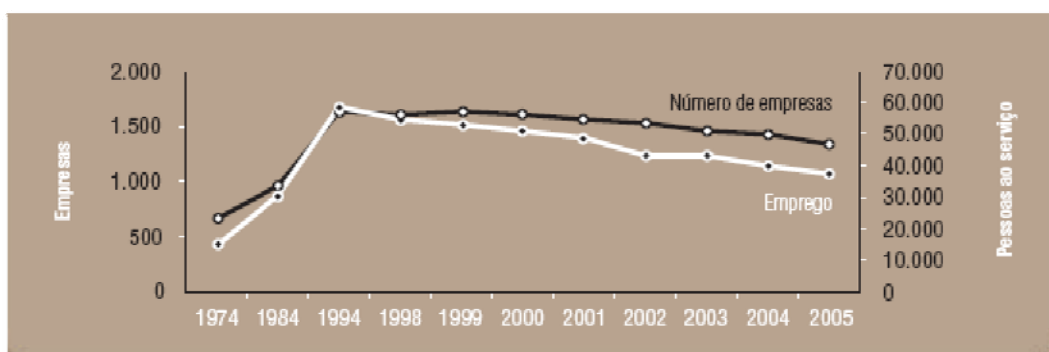
Figura 1.1 - Evolução da indústria portuguesa do calçado[1]

[1] Figura retirada de CEGEA (CEGEA, 2007)
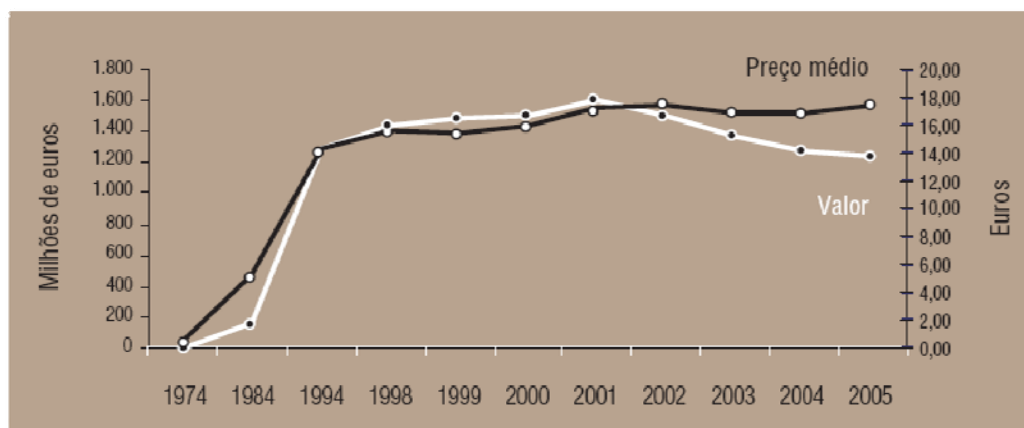
Figura 1.2 - Evolução das exportações portuguesas[2]

Consumidores

Actualmente a disponibilidade de diferentes tipos de calçado oferecida por grandes companhias de calçado é enorme, facultando ao consumidor um vasto campo de opções. Tal constitui um problema para as médias/pequenas empresas, visto que não têm a capacidade de acompanhar os requisitos do mercado como as empresas grandes (Spencer Jr, 1996).

Um importante factor também a considerar, é a moda. Esta influencia a forma como as empresas de calçado têm de responder às necessidades do mercado. Para as empresas de pequena/média dimensão se adaptarem, necessitam não só de manter qualidade e preços competitivos, mas também não esquecer a importância do consumidor como alvo do produto (Spencer Jr, 1996).

Robôs industriais na indústria do calçado

O fabrico de calçado em Portugal encontra-se muito dependente de processos manuais, exigindo uma elevada qualificação de mão-de-obra. Fruto desta dependência, ocorrem por vezes quebras elevadas de produtividade, causa da baixa flexibilidade apresentada pelos operadores. Isto acontece, entre outros motivos, porque é muito frequente numa empresa de calçado a introdução de novos modelos de sapatos, que leva a que seja necessário adoptar novos processos de manufactura e, consequentemente, obrigam a que se tenha de instruir devidamente os operadores para as suas novas funções. O processo de aprendizagem dos

[2] Figura retirada de CEGEA (CEGEA, 2007)

operadores não é instantâneo, uma vez que estes só atingirão o seu melhor desempenho após um período de adaptação. Estas limitações traduzem-se em quebras de produtividade e qualidade.

Simplificadamente, uma solução passaria pela introdução de sistemas automatizados numa empresa, no sentido de ajudar a combater a quebra no processo produtivo. No entanto, há que ter em conta outras variáveis, como por exemplo a complexidade apresentada pela maioria das operações na produção de calçado (como a costura e a montagem), que podem constituir um obstáculo. Para contornar os obstáculos e combater a quebra de produtividade, sugere-se a substituição dos processos que suscitem dificuldades na adaptação dos operadores, por processos automatizados.

Aplicações de robôs industriais na indústria do calçado

Até ao momento, não se conhecem processos inteiramente robotizados para a produção de sapatos. Apenas são encontradas na indústria do calçado aplicações específicas, onde se destacam (GMBH, 2007):

- A colagem;
- A pulverização;
- O desbaste;
- A cosedura das solas;

A realização destas operações por parte do robô, faz com que se verifique a eficiência do uso de materiais auxiliares (materiais de cementação[3], pulverização, etc.). Experiências realizadas pelo Departamento de Engenharia Mecatrónica da Universidade de Informação Tecnológica de *Tongmyong*, Coreia, demonstram que existe uma redução no uso dos materiais auxiliares em cerca de cinquenta porcento na maioria dos estudo, o que tem a vantagem de poupar ao nível dos custos e recursos (Kim, 2004). Outras vantagens dos sistemas robotizados são a rapidez de execução, trabalho de vinte e quatro horas e a possibilidade de trabalhar com químicos perigosos (Abreu, 2008). Nas desvantagens, pode referir-se que (Vilaça, et al., 2007): o resultado da execução depende do número de pontos

[3] Termo técnico para denominar a colagem.

escolhidos para a trajectória, a qualidade do processo depende da posição dos pontos escolhidos, o robô não faz a diferenciação das diferentes formas de calçado, tornando necessário o processo de o "ensinar".

Desta forma, quando um sapato tem de ser trabalhado, terá de se efectuar uma programação dedicada, isto é, terá de ser criado um programa ajustado à sua forma, número e orientação do pé. Isto obriga a que, para se obter alguma rentabilidade, se criem series para tratar o mesmo modelo, colocando o calçado no mesmo sítio para execução do mesmo programa. Um bom resultado dependerá, bastante, da experiência do operador encarregue do processo, visto que este terá de "ensinar" o robô por um de dois métodos: o *on-line*, ou o *off-line*. Assim, soluções apresentadas no âmbito da robótica podem constituir formas inovadoras de produzir calçado, que aplicadas inteligentemente podem favorecer a produção.

Processo de acabamento de solas duplas

Hoje em dia, devido ao avanço tecnológico existente, encontram-se na maioria do calçado solas produzidas por processos de injecção (o que é facilmente verificado, em qualquer sola com feitios mais elaborados) (ver Figura 1.3).

Figura 1.3 – Exemplo de calçado com sola procedente de injecção[4]

Todavia, sapatos com diferentes exigências de estética, conforto, qualidade, são elaborados através de outros processos. Num desses processos, a sola é composta por duas camadas, uma inferior e outra superior, como mostrado na Figura 1.4 (exemplo: sapato de vela).

[4] Imagem retirada de http://www.tonyshoes.com/images/footwear/ECCO/, visto em 20/03/09

Figura 1.4 – Exemplo de sapato composto por duas solas

No produto final, pretende-se que ambos os elementos constituintes da sola se assemelhem fisicamente, de modo a obter-se uma melhor apresentação do produto, bem como garantir uma maior resistência ao movimento.

Após contacto com uma entidade especializada (empresa Armipex[5]), verificou-se que, em média, a sola superior se salientava da inferior em cerca de um milímetro (Figura 1.5).

Figura 1.5 – Pormenor da diferença entre a sola superior e inferior antes do acabamento final

Na operação manual de uniformização da sola, existe um operador qualificado, que ao encostar e rodar o sapato, o mais uniformemente possível, sobre uma ferramenta abrasiva consegue realizar um bom acabamento (Figura 1.6).

Figura 1.6 - Processo de acabamento manual utilizando ferramentas abrasivas[6]

---

[5] Empresa sediada em S. Joões – Revinhade, Felgueiras, Portugal
[6] Imagem retirada de http://www.gettyimages.com/detail/200458971-001/Riser vista em 20/03/2009

1.1 Objectivos do Trabalho

O objectivo deste trabalho é o de conceber e simular uma célula robotizada para permitir o acabamento automatizado de sapatos de solas duplas, baseado na utilização de um robô industrial equipado com controlo de força. A célula robotizada deverá responder aos seguintes requisitos:

- Capacidade de acomodar diferentes modelos de sapatos;
- Capacidade de adaptação aos diferentes tamanhos de sapatos;
- Uniformidade do acabamento;
- Uma interface de comando intuitiva que permita uma fácil operação da célula por parte de um utilizador não especializado.

Para complementar o trabalho de simulação e sua aplicabilidade, serão efectuados testes utilizando o robô industrial *ABB IRB 2400*, disponível no laboratório de robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto.

1.2 Organização e Temas Abordados no Relatório

Tendo em conta os objectivos do trabalho, este documento está estruturado em sete capítulos, seguindo-se os anexos.

No primeiro capítulo é feita uma introdução ao tema e definidos os objectivos a alcançar.

No segundo capítulo, denominado de *Sistemas Robotizados na Indústria do calçado*, apresentam-se as diversas soluções robóticas existentes no mercado, assim como estudos efectuados relativos a soluções robotizadas para acabamento de solas duplas de sapatos.

Seguidamente, no capítulo três, *Estudo da Concepção da Célula Robotizada* faz-se uma abordagem inicialmente teórica, de modo a que seja possível reter conceitos importantes para a compreensão e estudo de células robotizadas. Posteriormente, são apresentados os vários estudos de soluções robotizadas referentes ao estudo do presente trabalho.

O *Sistema de Simulação e Programação Off-line* é descrito no quarto capítulo, que se inicia com uma contextualização sobre os tipos de programação de robôs existentes. Seguidamente, é introduzido o *software* de programação *off-line* utilizado, o *RobotStudio*, e são efectuadas algumas considerações sobre o seu uso.

No capítulo cinco, *Desenvolvimento da Simulação de uma Célula Robotizada*, é apresentado o processo de concepção e simulação de uma célula robotizada para realização do acabamento proposto neste trabalho. Neste capítulo, encontra-se também descrito todo o processo de desenvolvimento de uma aplicação gráfica para operação da célula robotizada.

O sexto capítulo, denominado de *Implementação da solução*, trata da verificação experimental do processo de acabamento de solas duplas em sapatos, de modo a ser possível obter conclusões acerca da aplicação robótica nesta área de trabalho.

No capítulo sete são tecidas conclusões, apresentando as limitações encontradas ao longo do desenvolvimento deste trabalho, e efectuadas sugestões para a realização de trabalhos futuros.

O documento faz-se acompanhar da respectiva bibliografia e dos anexos.

2 Sistemas Robotizados na Indústria do Calçado

A elaboração do presente trabalho teve como ponto de partida o estudo de soluções existentes para o processo de acabamento de solas de sapato. Como tal, foi realizada uma pesquisa exaustiva que permitisse contextualizar o trabalho desenvolvido e a pertinência da sua realização.

Este capítulo aborda a situação geral da robótica na indústria do calçado, sendo apresentadas e descritas soluções robotizadas oferecidas pelos diferentes fabricantes internacionais, bem como estudos efectuados que referem novas situações de implementação.

É necessário ter em consideração que ao longo deste trabalho, será utilizada tanto a palavra robô, como a expressão robô industrial. No entanto, ambas se referem a robô industrial. É relevante estabelecer esta diferença, devido às ambiguidades presentes em várias temáticas. No conceito forte da palavra robô no que denota à tecnologia, distinguem-se duas formas: uma pelos Estados Unidos da América e outra pelo Japão. Neste trabalho será utilizada a definição de acordo com o *Robot Institute of America*:

"A robot is a reprogrammable multifunctional manipulator designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks". (Hunt, 1983)

2.1 Sistemas Robotizados Comerciais

Os fabricantes de calçado têm reconhecido a necessidade de implementação de novas técnicas de fabrico, usufruindo de tecnologias de automação que possam garantir a viabilidade da execução dos diversos processos. Tal sucede que, tanto empresas, como instituições de

investigação, estão a desenvolver projectos de integração de sistemas robotizados para o fabrico de calçado.

Realizada uma pesquisa sobre os sistemas robotizados comerciais, verificou-se que existem quatro principais companhias a disponibilizar sistemas robotizados para a indústria de calçado: a *ACTIS Engineering*, a *DESMA*, a *Intelligent Machines Corporation* (*IMC*), e a *Robot System*.

2.1.1 *ACTIS Engineering*

A empresa francesa, *ACTIS Engineering,* desenvolveu uma aplicação automática, em colaboração com a companhia de robôs *Stäubli*, disponibilizando uma linha de produção que é comercializada como *Concept RB System* (Figura 2.1). Este sistema apresenta uma configuração em linha, constituído por um sistema de movimentação linear onde se encontram dispostas várias células robotizadas estrategicamente colocadas ao longo desta. A função destas células robotizadas é a de processar tarefas pré-programadas em partes do calçado (Kochan, 1996).
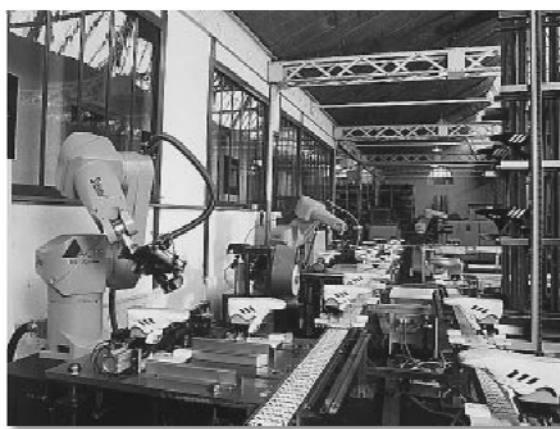
Figura 2.1 - Linha de produção de acabamento em calçado[7]

O sistema *RB* utiliza *software* próprio que controla o robô na célula e monitoriza a posição de cada calçado. Este sistema utiliza um método de identificação por etiquetas que contêm informações sobre o tamanho, estilo e orientação do sapato na célula. Isto permite que o sistema possa dar instruções ao controlador do robô para executar o carregamento do *software* apropriado a cada operação. O calçado é conduzido via palete instalada no sistema

[7] Figura retirada de Kochan (Kochan, 1996)

de movimentação linear, promovendo como tal, uma continuidade no sistema. A gestão das paletes é feita através da sua colocação em fila, de modo a que quando uma célula se torne disponível seja passível de ser imediatamente posicionada e processada. Cada célula utiliza um robô industrial equipado com ferramentas apropriadas para que possam executar tarefas, tais como, cementação e desbaste (Figura 2.2) (Kochan, 1996).
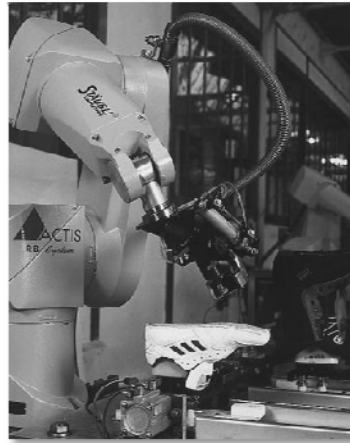
Figura 2.2 - Realização de uma operação de acabamento (*ACTIS Engineering*)[8]

2.1.2 DESMA

A *DESMA* é uma empresa alemã, que desenvolveu, em primeira mão no mercado, máquinas de moldação para manufacturar solas de calçado injectadas directamente sobre este. Esta solução é constituída pela utilização das máquinas de moldação (Figura 2.3) integradas com o uso de robôs industriais da *ABB* (Figura 2.4).
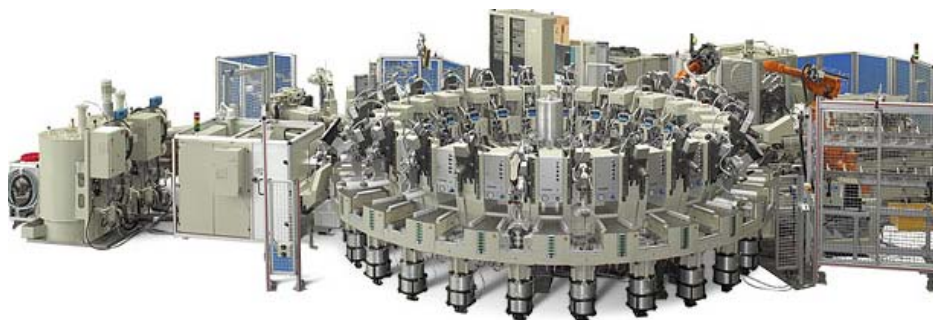
Figura 2.3 – Célula robotizada disponibilizada pela *DESMA*[9]

[8] Figura retirada de Kochan (Kochan, 1996)
[9] Imagem retirada de http://www.desma-china.com, visto em 22/03/09

Figura 2.4 – Robô *ABB* equipado com um sistema capaz de realizar acabamentos em calçado[10]

Os robôs industriais na célula robotizada apresentada anteriormente (Figura 2.3), têm como finalidade servir a máquina de moldação, sendo a sua utilização direccionada para a realização de acabamentos no calçado. As principais operações efectuadas por estes robôs são: o corte de excesso de material após moldação da sola (Figura 2.5), a cardagem (raspar a superfície onde será aplicada futuramente a cola, para que esta possa aderir melhor) (Figura 2.6), e a pulverização de superfícies com cola (Figura 2.7) (Spencer Jr, 1996).

Figura 2.5 - Operação de corte de excessos de material[11]

[10] Imagem retirada de http://www.desma.de/automation/automation.html, visto em 22/03/2009
[11] Imagem retirada de GMBH (GMBH, 2007)

Figura 2.6 – Operação de cardagem[12]
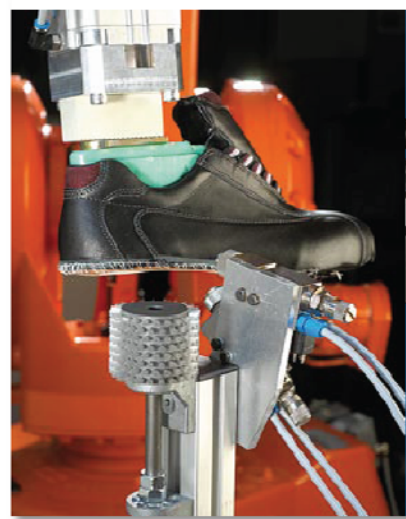
Figura 2.7 - Operação de pulverização de superfícies com cola[13]

De modo a permitir uma interacção simplificada com o operador, a *DESMA* desenvolveu uma aplicação para a consola do robô que permite a verificação gráfica da trajectória realizada pelo robô (Figura 2.8). Esta aplicação faz com que exista a possibilidade de alterar a programação do robô durante as operações no calçado, permitindo uma melhor obtenção de trajectórias de trabalho (Brorsson, et al., 2006).

Figura 2.8 - Interface gráfica da consola do robô *ABB*[14]

[12] Imagem retirada de GMBH (GMBH, 2007)
[13] Imagem retirada de GMBH (GMBH, 2007)
[14] Imagem retirada de GMBH (GMBH, 2007)

2.1.3 *Intelligent Machines Corporation (IMC)*

Esta empresa americana tem como principal objectivo criar soluções que consigam ultrapassar as maiores dificuldades sentidas na implementação de processos robotizados na indústria do calçado. Como tal, propõe sistemas dinâmicos que automaticamente se configurem/adaptem às operações a realizar no sapato. Este tipo de sistema inteligente oferece a possibilidade de realizar operações em condições imprevisíveis (como superfícies imprecisas). Esta companhia é detentora de uma patente que consiste na criação de calibres de contraste da forma do sapato. Esta utiliza uma tecnologia de preensão inovadora, permitindo que, quando um sapato é colocado na célula robotizada para execução de tarefas se possa colocar precisamente na mesma posição (Spencer Jr, 1996). A Figura 2.9 apresenta uma célula robotizada de demonstração desta empresa americana.

Figura 2.9 – Célula robotizada de demonstração da *IMC*[15]

2.1.4 *Robot System*

A *Robot System*, empresa italiana, tem como principal objectivo a criação de mecanismos automáticos e inovadores para a indústria do calçado. Esta entidade, no que remete a soluções robotizadas para realização de operações no calçado, dispõe de uma série de opções, recorrendo ao uso de robôs *Kawasaki's* de seis eixos. As principais operações disponibilizadas por esta empresa são (Robot System, 2009):

[15] Figura retirada de Kochan, (Kochan, 1996)

Cardagem - é efectuada por um sistema com uma cabeça especial de raspagem. Esta operação deverá ser realizada previamente à colocação de uma sola, permitindo que a fixação da sola possa ocorrer com sucesso (Figura 2.10).
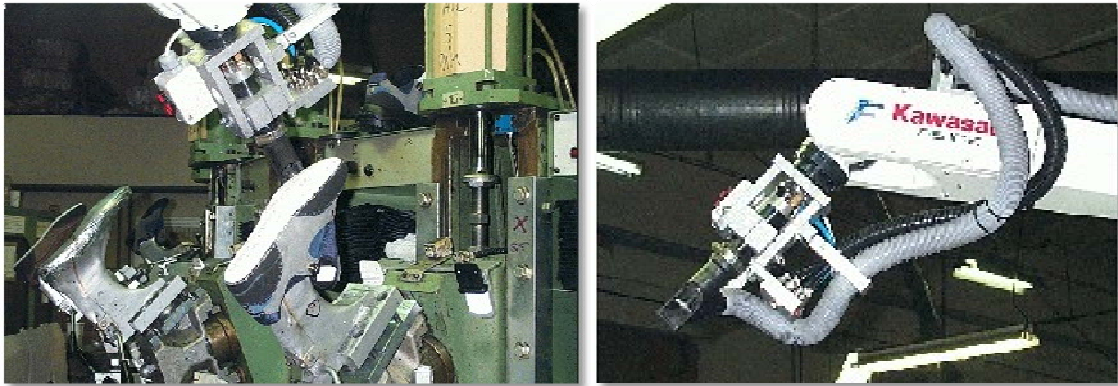
Figura 2.10 – Operação de cardagem (à esquerda) e respectiva ferramenta (à direita)

Extractor automático de jitos - consiste num manipulador de extracção de jitos de poliuretano (material utilizado nas solas) e limpeza da superfície do molde da sola injectada. Este manipulador é composto por uma pinça com movimento horizontal, para extracção do jito, e por uma lâmina com movimento vertical e horizontal para limpeza do molde (Figura 2.11).

Figura 2.11 – Sistema de extracção de jitos

Sistema de rotação de formas - constituído por um sistema reflector que lê a posição da forma, permitindo que as pinças se possam fixar correctamente nas formas para posterior rotação (Figura 2.12).

Figura 2.12 – Sistema de rotação de formas

Sistema de transporte de formas - é um sistema de transporte assíncrono utilizado para a interligação de postos de trabalho, quer robotizados, quer manuais (Figura 2.13).

Figura 2.13 – Sistema de transporte de formas

Sistema de reactivação de cola nas solas - constituído por um grupo de luzes, que através de um flash, garantem a reactivação da cola presente nas solas (Figura 2.14).

Figura 2.14 – Sistema de reactivação de cola nas solas

Identificação de formas - A identificação das formas dos sapatos é realizada automaticamente devido à presença de um transmissor-receptor instalado dentro da própria forma (Figura 2.15). A identificação é feita através de um código presente neste transmissor-receptor, que possibilita o reconhecimento da forma quanto aos seus atributos físicos (modelo, tamanho e orientação). Esta tecnologia de identificação usa um sistema de identificação de frequências de rádio (RFID), constituído por elementos transmissor-receptor colocados dentro das formas de sapato, uma antena exterior e uma unidade de leitura que comunicará com o computador, informando-o sobre as características das formas.

Figura 2.15 – Transmissor-receptor instalado na forma do sapato

De modo a facilitar a gestão e o controlo das operações nas indústrias, a *Robot System* desenvolveu um *software* denominado de *RS-WARE* (Figura 2.16), concebido especificamente para a utilização em robôs de seis eixos.

Figura 2.16 – Aspecto do *software RS-WARE*

Este *software* dispõe de uma interface intuitiva e simples, fornecendo ao operador uma simples forma de criação e alteração de um programa (Figura 2.17).
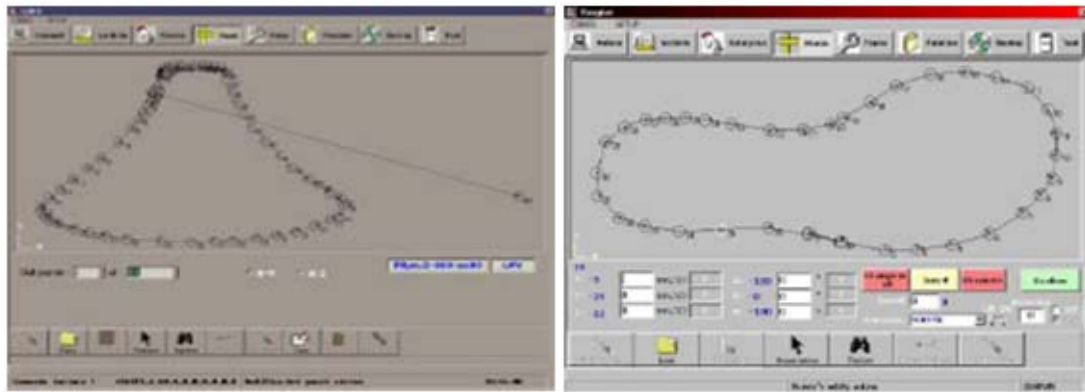
Figura 2.17 – Exemplo de alteração de um programa

Devido à necessidade de realizar programas para todas as formas de sapatos, este *software* vem equipado com uma função chamada *Grading* (Figura 2.18), que através de um modelo efectua um escalamento para todos os tamanhos requeridos pelo operador. Todas as operações efectuadas no *RS-WARE* poderão ser feitas em modo *off-line*, evitando assim paragens na produção.
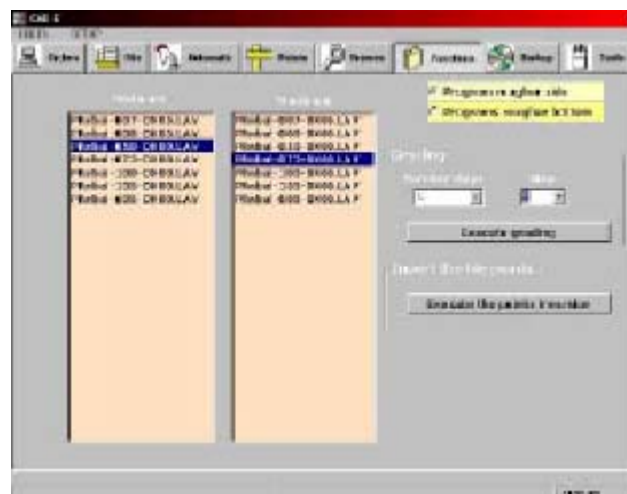
Figura 2.18 – Exemplo da operação de *Grading*

2.2 Sistemas Robotizados não Comerciais

No que diz respeito a sistemas robotizados não comerciais, tanto empresas, como Universidades estão à procura de novas soluções que consigam trazer o máximo de flexibilidade na produção de calçado, recorrendo ao uso de robôs industriais.

Muitos dos estudos recentemente efectuados, recorrem ao uso de uma imagem bidimensional, de modo a permitir obter apenas os pontos de interesse de um sapato, com o intuito de conseguir verificar uma solução de programação optimizada. Esta imagem pode ser adquirida de várias maneiras, no entanto as mais usuais são: o desenho em *CAD* e o uso de câmaras (Kim, 2004).

O processo de transformar as soluções propostas em programação é semelhante em ambos. No entanto, o modo de obtenção do desenho em *CAD* pelo uso de câmaras requer previamente um posto de processamento de dados apropriado, que permita identificar o calçado, conferindo a sua localização, orientação e perfil, tal como é sugerido na Figura 2.19.
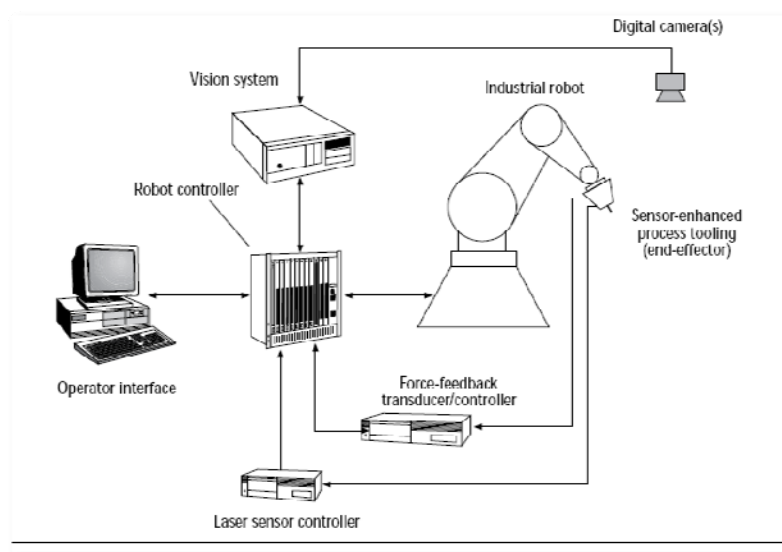
Figura 2.19 - Esquema de implementação possível com uso de câmara[16]

Seguidamente, ambos recorrem a um algoritmo apropriado que processa os dados adquiridos de modo a disponibilizar informação para o controlador do robô. A informação que terá de ser enviada ao controlador é dependente da tarefa a executar. No entanto, as aplicações mais estudadas são o desbaste e a colagem. Como tal, o algoritmo terá de extrair um contorno da sola e da sua posição (bidimensional ou tridimensional, dependente da

[16] Figura retirada de Spencer Jr (Spencer Jr, 1996)

complexidade da operação e do algoritmo). Depois será enviada, ao controlador, a informação pós-processada que deve conter informação da posição e das coordenadas reais da sola, bem como do posicionamento a dar à ferramenta de trabalho. Assim, este processo estará em condições de ser iniciado contendo informações da trajectória a tomar. Ao analisar cada objecto individualmente, este tipo de sistemas confere uma forma expedita de realizar operações, conferindo flexibilidade ao sistema (Vilaça, et al., 2007).

2.3 Conclusões

A pesquisa de soluções robotizadas para efectuar acabamentos em sapatos revelou-se numa tarefa árdua, dada a escassez de informação existente (essencialmente em suporte digital). Os principais aspectos a concluir sobre este levantamento de soluções robotizadas são:

Os processos de acabamento apresentados denotam a existência de uma pré-programação dos robôs, ou seja, é necessário conhecer previamente o calçado que irá ser processado nas células robotizadas.

A identificação do calçado nas células robotizadas é realizada através de sistemas com capacidade de obter informações sobre o tipo, tamanho e orientação do calçado. Os sistemas de identificação encontrados foram sistemas de identificação por etiquetas ou, de uma forma mais avançada, por tecnologia sem fios.

Na à monitorização da célula robotizada é comum existir *software* dedicado que permite, não só a verificação do decorrer dos processos, como também a optimização do processo através das funcionalidades disponíveis.

Relativamente a células robotizadas usufruindo da tecnologia de controlo de força, não foram encontradas quaisquer referências, talvez por se tratar de uma tecnologia recente e em expansão.

3 Estudo da Concepção da Célula Robotizada

Este capítulo estuda os vários pontos essenciais à concepção de uma célula robotizada. Para esse efeito, é elaborada uma contextualização teórica para uma melhor compreensão do que se pretende estudar.

3.1 Conceito de Célula Robotizada

Nos tempos que correm, os robôs apresentam uma alargada gama de capacidades, possibilitando a realização de aplicações industriais. No entanto, para que seja realizada uma dada operação, são necessários sistemas auxiliares: sistemas de movimentação linear/angular, ferramentas, máquinas de produção e produtos. Para que se torne possível a coordenação dos vários equipamentos com as actividades a realizar, é indispensável recorrer ao estudo do *layout* da célula robotizada (McKerrow, 1991).

Uma célula robotizada é descrita como uma área de trabalho devidamente estudada com um objectivo de produção comum, onde existem distribuições de tarefas, interligando equipamentos, na qual estará um ou mais robôs, permitindo assim a optimização e realização da(s) operação(ões).

3.2 Principais Sistemas Constituintes de uma Célula Robotizada

Os principais sistemas que uma célula robotizada deve possuir são, nomeadamente (Abreu, 2001):

- **Sistemas de alimentação/remoção de produtos -** são aqueles que vão permitir a alimentação do produto, onde, após operação, terão de ser retirados por um outro sistema, o de remoção;

- **Sistemas de posicionamento do produto -** referem-se à colocação do sapato no sítio apropriado para que possa ser realizada a operação de acabamento;

- **Sistemas de segurança -** referem-se a sistemas que garantem a protecção do equipamento e operador;

- **Sistema de supervisionamento de operações** - poderá ser realizado através de um computador que visione o decorrer do processo, indicando ao operador o ponto de situação dos vários estados da célula, ou o próprio operador poderá supervisionar visualmente;

- **Robô industrial com respectiva ferramenta** - ferramenta que forma o principal conjunto, uma vez que será este o conjunto a realizar a tarefa principal, o acabamento;

- **Sistema de inspecção** - será realizado por um sistema que possa comparar o resultado com um calçado de referência, ou será uma inspecção visual realizada por um operador.

3.3 Configurações das Células Robotizadas

As células robotizadas podem ser organizadas em várias configurações. Estas podem ser classificadas em três tipos, seguidamente apresentadas:

- Robô centrado na célula;
- Robô em linha na célula;
- Robô móvel na célula.

3.3.1 Robô Centrado na Célula

Nesta configuração o robô localiza-se no centro da célula e o equipamento auxiliar é colocado à sua volta em forma circular, como ilustrado na Figura 3.1. Tipicamente, nesta configuração, o robô realiza quer a operação de produção, quer a tarefa de alimentação/remoção. Como exemplo para este tipo de configuração existem aplicações por parte das indústrias de fundição, onde o robô se encarrega de remover o produto do molde (depois de completo o ciclo de fundição) e, posteriormente, o leva para um banho de têmpera, movimentando assim a peça duma operação para outra (Hunt, 1983).
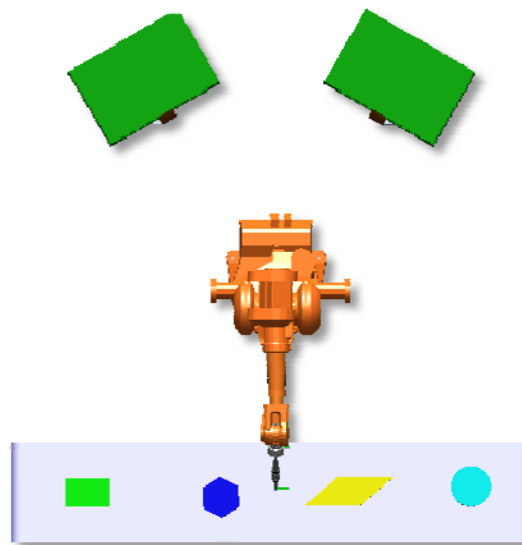
Figura 3.1 – Robô centrado na célula

3.3.2 Robô em Linha na Célula

Nesta configuração o robô é localizado ao longo de uma linha de produção, na qual executa uma tarefa pré-destinada assim que lhe chegar o produto. Nesta configuração é normalmente usado mais do que um robô na mesma linha de movimentação. Esta configuração pode ser observada na Figura 3.2 (Groover, 1996).
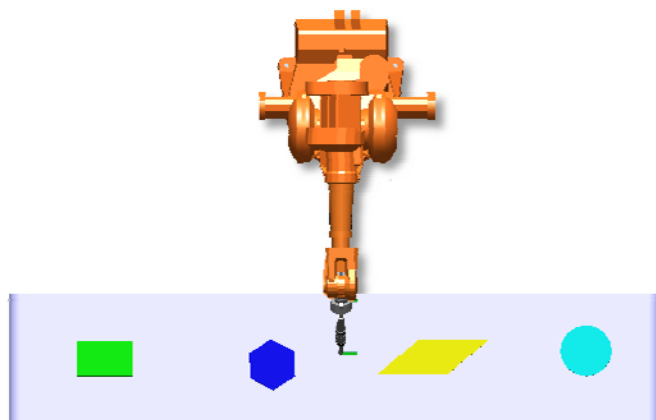
Figura 3.2 – Robô em linha na célula

3.3.3 Robô Móvel na Célula

Na configuração apresentada na Figura 3.3, o robô tem capacidade de se mover dentro da célula, para os vários postos de trabalho. Esta solução de configuração é normalmente feita com a disposição de um robô numa base móvel, que poderá ser constituída por um sistema de carris, que podem ou não estar suspensos. Este tipo de configuração torna-se útil quando o robô tem de servir várias máquinas de produção com ciclos de produção elevados. O problema de design associado a esta configuração é a determinação do número de máquinas de produção para cada robô, dado que o objectivo é não ter, dentro do possível, qualquer tempo morto nas máquinas de produção (Hunt, 1983).
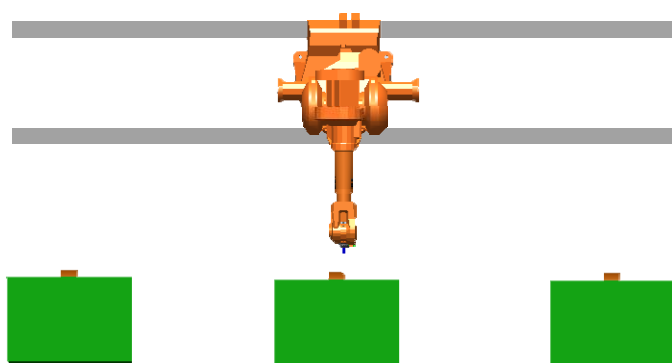
Figura 3.3 – Robô móvel na célula

Os diferentes tipos de configurações apresentadas previamente são essenciais quanto à sua possível implementação em fábricas, podendo estas células incorporar parte do *layout* da fábrica (Koren, 1987).

3.4 Sistemas de Transporte nas Linhas de Produção

Os sistemas de transporte são utilizados no processo de produção em série para que o produto/material a transformar possa ser movimentado ao longo dos postos de trabalho devido à presença de robôs e/ou outras entidades de ordem mecânica, ou humana (Abreu, 2001). Podem ser encontrados nas linhas de montagem os seguintes tipos de sistemas de transporte:

- Sistema de transporte intermitente ou síncrono;
- Sistema de transporte contínuo;
- Sistema de transporte assíncrono.

3.4.1 Transporte Intermitente ou Síncrono

Consiste num transporte em simultâneo, mas intermitente, dos produtos entre os vários postos, ou seja, consiste hipoteticamente num sistema "pára-arranca" sincronizado. O robô neste sistema é mantido numa posição estacionária onde aguarda a chegada do produto. A vantagem que este sistema oferece é a garantia que, quando o robô tem de efectuar uma operação no produto, este se encontra fixo e orientado, facilitando a sua programação.

3.4.2 Transporte Contínuo

Como o nome sugere, este sistema oferece um transporte contínuo dos produtos com uma velocidade constante. Ao contrário do transporte intermitente, este levanta problemas na constante variação da posição e orientação dos produtos. No entanto, este pode ser controlado de duas formas:

- *Com sistemas de guias que permitam mover o robô* - o robô irá realizar um movimento paralelo à linha de produção, de modo a conseguir obter uma velocidade relativa constante que permita executar a tarefa em movimento.

- *Com o robô estacionário* - esta situação requer que o robô tenha recursos computacionais suficientes, que o permitam adaptar-se constantemente às novas coordenadas dos produtos que se encontram em movimento na linha de produção.

3.4.3 Transporte Assíncrono

Este sistema caracteriza-se pelo facto de apenas ser movimentado um produto após realizada uma operação, ou seja, assim que uma operação é terminada, inicia-se o movimento para a próxima área de trabalho. O problema inerente a este tipo de sistema é o facto de exigirem movimentações independentes entre cada área de trabalho.

3.5 Estudos de Diferentes Células Robotizadas

O estudo da célula robotiza para proceder à realização de uma tarefa de acabamento, consiste, nesta primeira fase, em analisar as várias configurações possíveis para implementação. É necessário então, avaliar o número de robôs necessários e a sua colocação face às possíveis formas de alimentação e remoção do sapato e cadências de trabalho pretendido. Existe também a possibilidade de se considerar que o robô e/ou o sapato sejam móveis. No entanto, a possibilidade do robô se mover numa linha própria, foi desde logo posta de parte, face às características apresentadas pelo sapato (pequeno e leve) e, ao facto de representar um custo de implementação acrescido, desnecessário para este processo de trabalho.

Seguidamente, são sugeridas algumas configurações de implementação e respectivas ilustrações elucidativas.

3.5.1 Estudo de Células Robotizadas com um Robô

1. Robô disposto em Linha com o Sistema de Movimentação Linear

Nesta configuração, ilustrada na Figura 3.4, o robô executa o acabamento no sapato, colocado numa posição previamente definida. Esta linha deve funcionar com um movimento assíncrono, de modo que quando o robô termine a sua operação, possa receber o próximo sapato. Um sistema de alimentação e remoção é encarregue de realizar o carregamento e descarregamento dos sapatos na linha nos tempos do acabamento realizados por parte do robô.
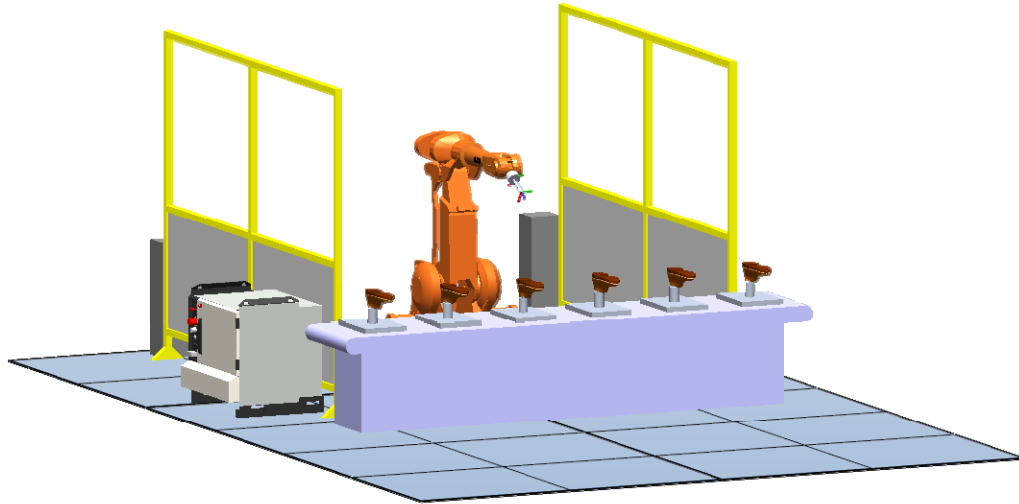
Figura 3.4 – Exemplo do robô em linha com sistema de movimentação de sapatos

2. Robô disposto em Linha com o Sistema de Movimentação Angular

Na seguinte configuração, ilustrada na Figura 3.5, o robô executa o acabamento no sapato, colocado numa posição previamente definida. Esta mesa deverá funcionar como a configuração anterior, com movimento assíncrono pelas mesmas razões. Nesta situação, um sistema de alimentação e remoção terá de realizar o descarregamento do sapato pronto e, consequente, carregamento de um outro por acabar, nos tempos de execução de operação por parte do robô. O número de sapatos presentes na mesa está dependente do tamanho da mesa e respectivo número de posições.
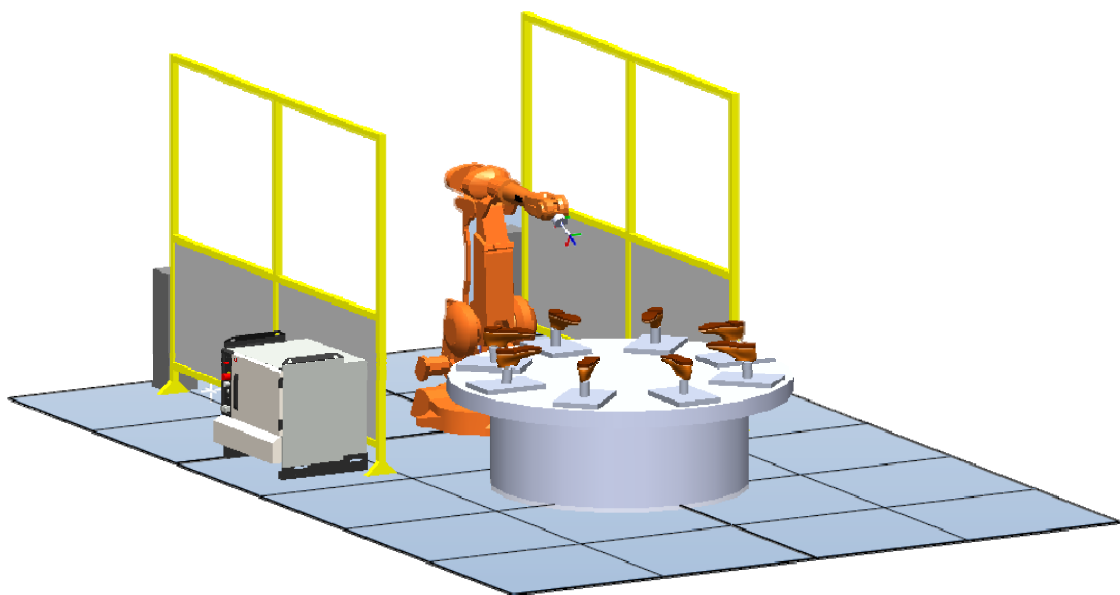
Figura 3.5 – Exemplo de robô disposto em linha com o sistema de movimentação angular

3. Robô Centrado com a Linha de Movimentação Linear e a Máquina Dedicada

Na Figura 3.6, está apresentada uma solução, em que o robô se encontra a operar centrado na célula. Nesta situação, o robô tem como função, pegar no sapato da presente linha, movimentar-se para a máquina de acabamento, executar o acabamento e regressar à linha para recolocação do sapato. Nesta configuração é utilizado um movimento assíncrono.
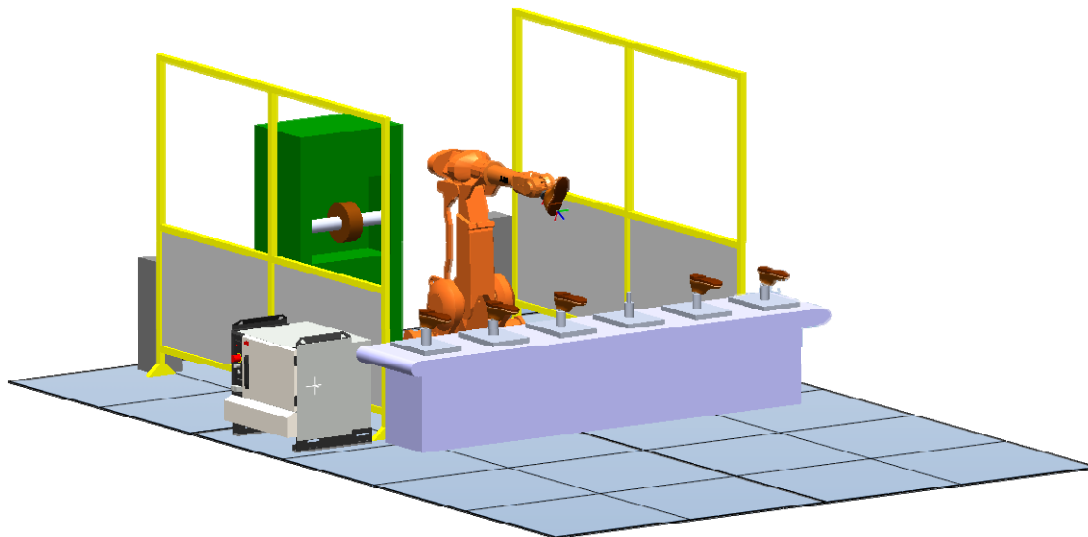
Figura 3.6 – Exemplo de robô centrado entre a linha de movimentação e a máquina dedicada

Para todas estas configurações apresentadas serão necessários os seguintes recursos:

- Robô e respectivo controlador;
- Sistema de movimentação linear/angular;
- Sistema de alimentação e remoção;
- Ferramenta apropriada à realização do acabamento;
- Máquina de acabamento (exclusivamente para a configuração 3);
- Suportes para os sapatos presentes no sistema de movimentação;
- Sistema de segurança adequado.

3.5.2 Estudos comparativos entre as várias configurações propostas

Com vista a poder decidir sobre a configuração a implementar, é necessário estabelecer comparações entre as configurações anteriormente apresentadas, consoante alguns parâmetros. A Tabela 3.1 pretende ilustrar essa comparação.

Nº	Interligação directa com outros postos de trabalho	Tempo de operação do robô	Tempos de operação dos sistemas de alimentação e remoção	Contaminação dos sistemas de movimentação com lixo resultante do processo de acabamento
1	Possível	2 Ciclos	1 Ciclo	Existente
2	Não é possível	2 Ciclos	2 Ciclos	Existente
3	Possível	4 Ciclos	1 Ciclo	Não existente

Tabela 3.1 - Comparação entre as configurações das células robotizadas

De maneira a elucidar sobre os dados presentes na Tabela 3.1, irá ser feita uma breve descrição sobre alguns dos parâmetros considerados.

A interligação directa com outros postos de trabalho

Mostra a possibilidade que a célula robotizada tem de se interligar com outros postos de trabalho. O facto de nas células número 1 e 3 ter sido considerado como possível, deve-se à sua configuração em linha, nas quais o sistema de movimentação linear poderá incorporar outras operações na mesma linha. A impossibilidade na célula número 2 deve-se à existência de um sistema de movimentação angular que, ao funcionar rotativamente, limita a transferência para outros postos, uma vez que este apenas poderá operar para si próprio.

Quanto aos tempos existentes, estes podem ser distinguidos em dois grupos: os tempos de operação do robô e os tempos de alimentação/remoção.

Tempos de operação do robô

De acordo com a Tabela 3.1, os ciclos de operação do robô para as configurações 1 e 2 são:

- Avanço por parte do robô para realizar a operação de acabamento no sapato;
- Recuo do robô após operação e simultâneo avanço do sistema de movimentação.

No caso 3, os ciclos são:

- Avanço do robô até ao sistema de movimentação linear para agarrar o sapato;
- Avanço do robô até à máquina de acabamento com consequente operação no sapato;
- Avanço novamente até ao sistema de movimentação linear para recolocação do sapato;
- Recuo do robô com avanço simultâneo do sistema de movimentação linear.

Tempos de operação dos sistemas de alimentação e remoção

Os ciclos de operação dos sistemas de alimentação e remoção, para a configuração 1 e 3, são os seguintes:

- Alimentação e remoção do sapato simultânea, durante o processar de um sapato.

No caso número 2, o sistema de alimentação e remoção realizará dois ciclos constituídos pela remoção do sapato pronto e pela alimentação de um novo por acabar.

3.5.3 Estudo de Células Robotizadas com mais de um Robô

No tópico anterior as soluções apresentadas retratam situações que utilizam um robô. Todavia, é possível incorporar mais robôs para cada uma das configurações propostas.

1. Robôs em Linha com o Sistema de Movimentação Linear

Na Figura 3.7, semelhante à configuração 1 do tópico 3.5.1, está ilustrada a presença de dois robôs. Contudo, poder-se-á incorporar tantos robôs, quantos os necessários. A finalidade da existência de um maior número de robôs nesta configuração é proporcionar a realização de diferentes tipos de acabamentos de um modo assíncrono.
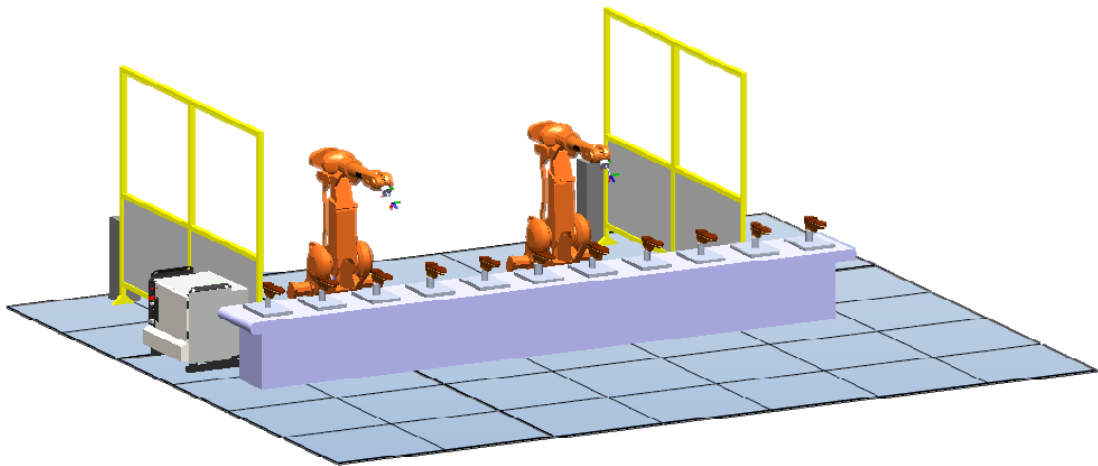
Figura 3.7 – Exemplo de robôs em linha com o sistema de movimentação linear

2. Robôs em Linha com o Sistema de Movimentação Angular

A Figura 3.8 ilustra o uso de dois robôs. Nesta configuração, poderão ser realizadas duas operações de acabamento diferentes, para o caso de só existir um sistema de alimentação e remoção, que retira o sapato acabado e coloca um novo durante a operação dos robôs.

No caso de existirem dois sistemas de alimentação e remoção, a produção de sapatos com acabamentos realizados é dupla. Cada um destes sistemas terá como função retirar o sapato acabado e colocar um novo, quando o robô se encontra em funcionamento. O número de sapatos presentes na mesa depende apenas da dimensão da mesa.
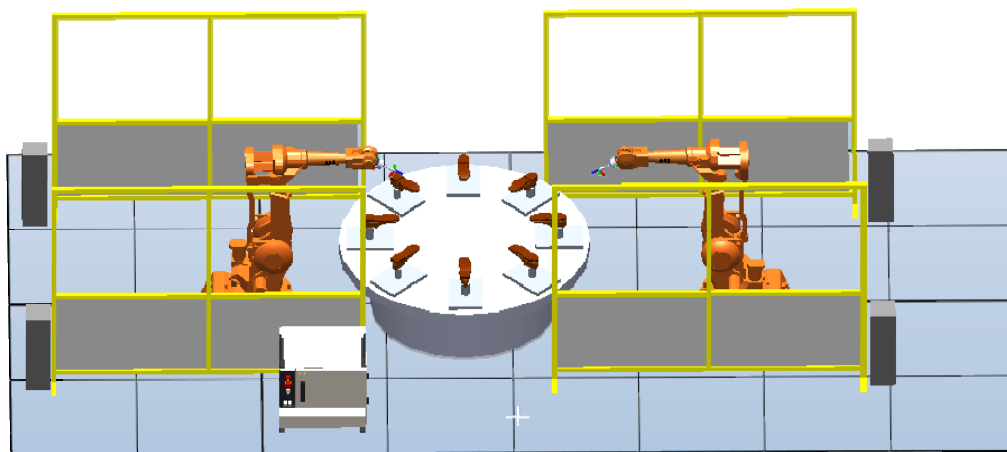
Figura 3.8 - Exemplo de robôs em linha com o sistema de movimentação angular

3. Robô Centrado entre a Linha de Movimentação Linear e as Máquinas Dedicadas

Na Figura 3.9, está apresentada uma solução em que o robô se encontra a operar centrado na célula. Nesta situação, o robô tem como função pegar no sapato da presente linha e movimentar-se, sequencialmente, sobre as várias máquinas de acabamento existentes. No fim do processo, deverá regressar à linha para recolocação do sapato.
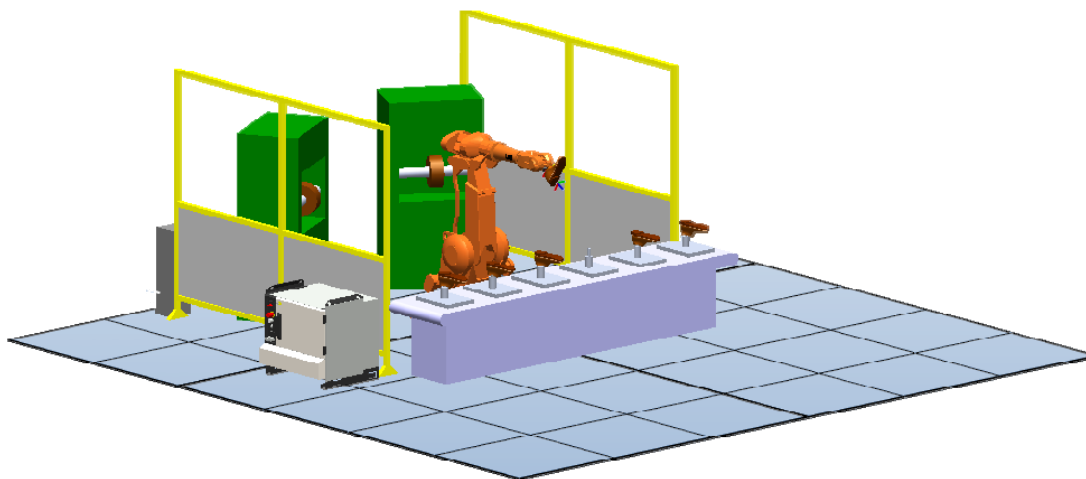
Figura 3.9 - Exemplo de robô centrado entre a linha de movimentação e as máquinas dedicadas

As soluções apresentadas são meramente estudos que denotam a possibilidade de incorporação de células robotizadas no ambiente fabril para produção de calçado. A escolha de uma das configurações apresentadas dependerá do *layout* da empresa e dos equipamentos que incorpora, daí não ter sido feita nenhuma observação sobre qual será a melhor solução para a indústria do calçado.

3.6 Abordagem para Aplicação Robótica

A adopção de uma das soluções propostas pode ser realizada com o intuito de substituir um determinado processo/operação ou para a introdução de um novo. Em ambos os casos deve-se proceder à elaboração de um plano de instalação, que permita a incorporação de um robô na linha de produção em causa.

É de facto importante que antes da adopção de uma solução robotizada se faça um levantamento do plano de operações em causa e se proceda à comparação de um robô industrial com uma máquina dedicada, de modo a estudar a possível aplicabilidade e rentabilidade de uma solução robotizada.

3.6.1 Planeamento de Operações

No planeamento de uma operação deve começar por definir-se o *output* do sistema. É indispensável questionar o que terá o sistema de realizar, de modo a ser possível estabelecer os *inputs* necessários à concretização da operação pretendida. Posto isto, deve-se construir uma solução, a partir das acções (*output*) para as medições (*input*).

No conceito deste trabalho, estabeleceu-se uma decomposição do planeamento, do seguinte modo, baseado no autor McKerrow (McKerrow, 1991):

- *Acção* - O robô deve executar uma operação de acabamento num sapato específico, presente numa linha de produção. Como tal, deve identificar o sapato, localizar a sua posição e orientação.
- *Planeamento de acções* - Deve decompor-se em sequências repetitivas de acções: mover para a posição inicial, identificar o objecto, mover a ferramenta até ao objecto, executar o acabamento, afastar-se para uma posição de referência.
- *Percepção* - Torna-se indispensável que o controlador do robô tome decisões com base na informação recebida pelos sistemas sensoriais, uma vez que existem sapatos de diferentes tipos e gamas.
- *Modelação* - Para atingir o objectivo em causa, é indispensável que a informação recebida pelos sensores seja reunida num modelo de operação, que

irá ser utilizado pelo robô, com vista a reconhecer o sapato. Consequentemente, é calculado o modo como se procede a operação.

- *Medição* – É necessário fazer a identificação do sapato, de modo a determinar-se qual a sua forma, tamanho e orientação do pé.

3.6.2 Comparação Robô Industrial *vs* Máquina dedicada

A comparação entre um robô industrial e uma máquina dedicada deve ser realizada de acordo com parâmetros que incluam o operador, a empresa e o consumidor. É importante estabelecer estas comparações uma vez que deve ter-se em atenção o porquê de uma mudança de método de produção.

No corrente trabalho, foi efectuada uma possível comparação entre um robô industrial e uma máquina dedicada, para o processo de acabamentos de solas de calçado. Este estudo encontra-se esquematizado na tabela apresentada seguidamente (Tabela 3.2).

	Robô Industrial	**Máquina Dedicada**
Rapidez de execução	Constante em todos os ciclos de operação, executando à velocidade máxima permitida nas condições satisfatórias à aprovação.	Dependência do operador.
Qualidade do produto	Todos os produtos terão a mesma qualidade.	Dependência do operador.
Segurança oferecida ao operador	Maior, dado que trabalha num área protegida.	Menor, uma vez que trabalha com a máquina.
Disponibilidade do operador	Maior disponibilidade para realização de outras tarefas.	Intensidade diária de operação.
Quantidade produzida	Constante	Dependência do operador.
Custo do equipamento	Maior investimento inicial.	Menor investimento inicial.

Tabela 3.2 - Comparação Robô vs Máquina Dedicada

Por motivos de simplificação, as comparações efectuadas na Tabela 3.2 são consideradas teóricas, quer isto dizer que as condições de operação serão vistas como sendo perfeitas por parte dos equipamentos (ignorando possíveis desgastes de ferramentas, falhas dos sistemas mecânicos, entre outros).

A dependência no operador denota que, ao longo do seu horário laboral, este, perde capacidades de trabalho. Isto deve-se à repetibilidade das operações, que aliados ao cansaço físico e cansaço psicológico fazem baixar a produção.

4 Sistema de Simulação e Programação *Off-line*

No presente capítulo é feita uma breve contextualização sobre os tipos de programação de robôs existentes. Posteriormente, é explicado o princípio de funcionamento do *software* de concepção e simulação interactiva utilizado, o *RobotStudio*, e demonstrada a metodologia adoptada, desde a modelação tridimensional até à simulação de operações.

4.1 Programação de Robôs

A programação dos robôs, nos seus primórdios, começou por ser efectuada na linha de produção, requerendo a utilização do robô. Cada vez que fosse necessário realizar modificações à programação do robô, era necessário parar a produção da célula robotizada onde este se inseria. Este tipo de programação designa-se por programação *on-line* (Hunt, 1983).

Actualmente, devido à forte evolução tecnológica, existem processos de simulação e animação gráfica, baseados em computador, que permitem facilitar a programação dos robôs. Esta programação, por não recorrer ao uso directo do robô, é denominada de *off-line*, e tem vindo a ser utilizada com uma frequência cada vez maior, devido não só às suas potencialidades, mas também por todos os fabricantes de robôs a disponibilizarem.

4.1.1 Programação *On-line*

A programação denominada de *on-line*, é um método de programação que envolve o uso directo do robô para a sua implementação (daí o termo em inglês, *on-line*).

O princípio desta metodologia de programação implica que o programador movimente o robô até às posições pretendidas, gravando posteriormente cada uma das instruções na memória do controlador, constituindo no final, o programa desejado.

A realização desta programação é conseguida sobre duas formas:

- A primeira forma consiste em mover o robô até ao sítio pretendido, interagindo fisicamente com este, ou seja, pegando literalmente no punho do robô, movimentando-o até ao sítio pretendido. Esta programação é utilizada quando existem trajectórias complexas, como é o caso da aplicação de tinta por *spray*, uma vez que é fundamental a existência de uma trajectória que garanta a aplicação da tinta de um modo uniforme.
- A segunda forma consiste em movimentar o robô, utilizando a sua consola de interface, denominada no termo técnico de *teach-pendant*. Esta ferramenta é normalmente utilizada quando não existem trajectórias complexas, ou quando não é necessária grande precisão. Como exemplo, tem-se a movimentação de peças dum sítio para outro *(pick-and-place)*, onde uma boa aproximação dos pontos poderá originar uma solução satisfatória.

O uso da programação *on-line*, por qualquer uma das formas de programação apresentadas, traduz-se num método relativamente simples de programar. Contudo, o uso directo do robô, obriga a que se tenha forçosamente de parar a célula robotizada originando quebras de produtividade. Outro facto decorrente do uso directo do robô é o risco de ocorrerem colisões entre os vários equipamentos presentes na célula ou até mesmo perigo de lesão humana, durante a fase de programação do robô.

4.1.2 Programação *Off-line*

A programação *off-line*, de modo contrário à metodologia apresentada anteriormente, não recorre ao uso directo do robô.

Esta metodologia de programação do robô consiste na inserção de linhas de comandos sucessivos numa linguagem própria, recorrendo quer ao uso de *software* específico, quer ao uso de programas de edição de texto.

Este método de programação é cada vez mais utilizado, por possibilitar o desenvolvimento de um programa sem proceder à paragem da célula robotizada. Contudo, este método torna-se complicado de implementar uma vez que os pontos de movimento definidos textualmente não trazem consigo uma visualização gráfica. Assim, após desenvolvimento do programa, dever-se-á testá-lo *on-line*, para depurar qualquer tipo de erro existente face à definição dos pontos de movimento.

De modo a contornar as dificuldades sentidas na programação *off-line*, enunciadas acima, podem ser utilizadas soluções de *software* de programação *off-line* com ferramentas gráficas, que permitem traduzir a programação textualmente desenvolvida numa simulação gráfica ilustrativa.

4.1.3 Programação *Off-line* Interactiva

A programação *off-line*, aliada à simulação gráfica interactiva, é cada vez mais utilizada nas indústrias nos dias de hoje, devido às suas enormes potencialidades.

O *software* permite o desenvolvimento de programas interagindo com o ambiente gráfico, onde, por exemplo, é possível movimentar o modelo do robô para criar um programa e verificar de um modo imediato a sua possibilidade de aplicação. Este tipo de *software*, torna-se de tal forma poderoso que possibilita:

- Transferir para o controlador do robô os programas efectuados, sem ser necessário realizar correcções (quando existe uma calibração adequada);
- Verificar as trajectórias programadas, minimizando os erros de programação;
- Dispensar a paragem da célula robotizada.

Na Tabela 4.1 listam-se, não exaustivamente, alguns produtos existentes no mercado de *software* de programação *off-line* interactiva. É cada vez mais usual cada fabricante de robôs disponibilizar a sua própria aplicação.

Fornecedor	Nome do Software	Versão actual do software
ABB Robotics	*RobotStudio*	5.11.02
Fanuc	*RoboGuide*	-
Kuka	*Kuka Sim*	2.1
MotoMan	*MotoSim EG*	2.10
Robot Simulations	*WorkSpace*	5
TecnoMatix	*RobCad*	-

Tabela 4.1 - Exemplo de *Software* de programação *Off-line* existente no mercado[17]

4.2 *RobotStudio* da *ABB Robotics*

No âmbito deste trabalho, irá ser utilizado o *software* de programação *off-line* da *ABB Robotics*, denominado de *RobotStudio*. Este *software* utiliza a linguagem de programação *RAPID* da *ABB*.

Neste tópico, é feita uma abordagem generalista sobre o *RobotStudio* e a metodologia a utilizar até chegar à simulação do acabamento de sapatos.

4.2.1 Capacidades do *RobotStudio*

A *ABB Robotics* criou um novo conceito de programação, que denominou por *True Off-line Programming*, por permitir simular o funcionamento do controlador real do robô dentro da sua própria aplicação de programação *off-line, RobotStudio*. (ABB, 2002)

Esta ferramenta revela-se sem dúvida bastante útil no desenvolvimento de programas para o robô, permitindo que seja possível:

- Importação de ficheiros *CAD*;
- Geração de trajectórias automaticamente a partir de um modelo *CAD*;

[17] Visto em 15/04/09

- Optimização de trajectórias (indicando possíveis pontos de singularidade);
- Detecção de colisões;
- Verificação de alcance;
- Adição de pacotes de aplicações extra (ex: aplicações para maquinagem, soldadura, etc.);
- Verificação tridimensional das trajectórias programadas;
- Verificação da aplicabilidade do programa na célula real;[18];
- Introdução imediata de um programa desenvolvido no controlador real do robô, sem qualquer manipulação posterior de código[19].

4.2.2 Estrutura de Programação do *RobotStudio*

A estrutura do *software* de programação *RobotStudio* consiste na interacção de diferentes módulos, que vão desde uma modelação tridimensional de componentes, até ao programa a descarregar para o robô.

De modo a compreender o funcionamento destes módulos e a sua integração, permitindo o processamento de informações até à obtenção do programa final, refez-se a estrutura de programação proposta pelo autor Chan (Chan, et al., 2003), com as respectivas alterações que se adequam ao *RobotStudio*. Esta nova estrutura encontra-se ilustrada na figura seguinte (Figura 4.1).

[18] Devido à presença do controlador virtual, que realiza cálculos de cinemática inversa, garantindo de certa forma, que se o programa funcionar na simulação, irá certamente funcionar na aplicação real.
[19] Este facto depende do grau de reprodutibilidade da célula real para a virtual (calibração).
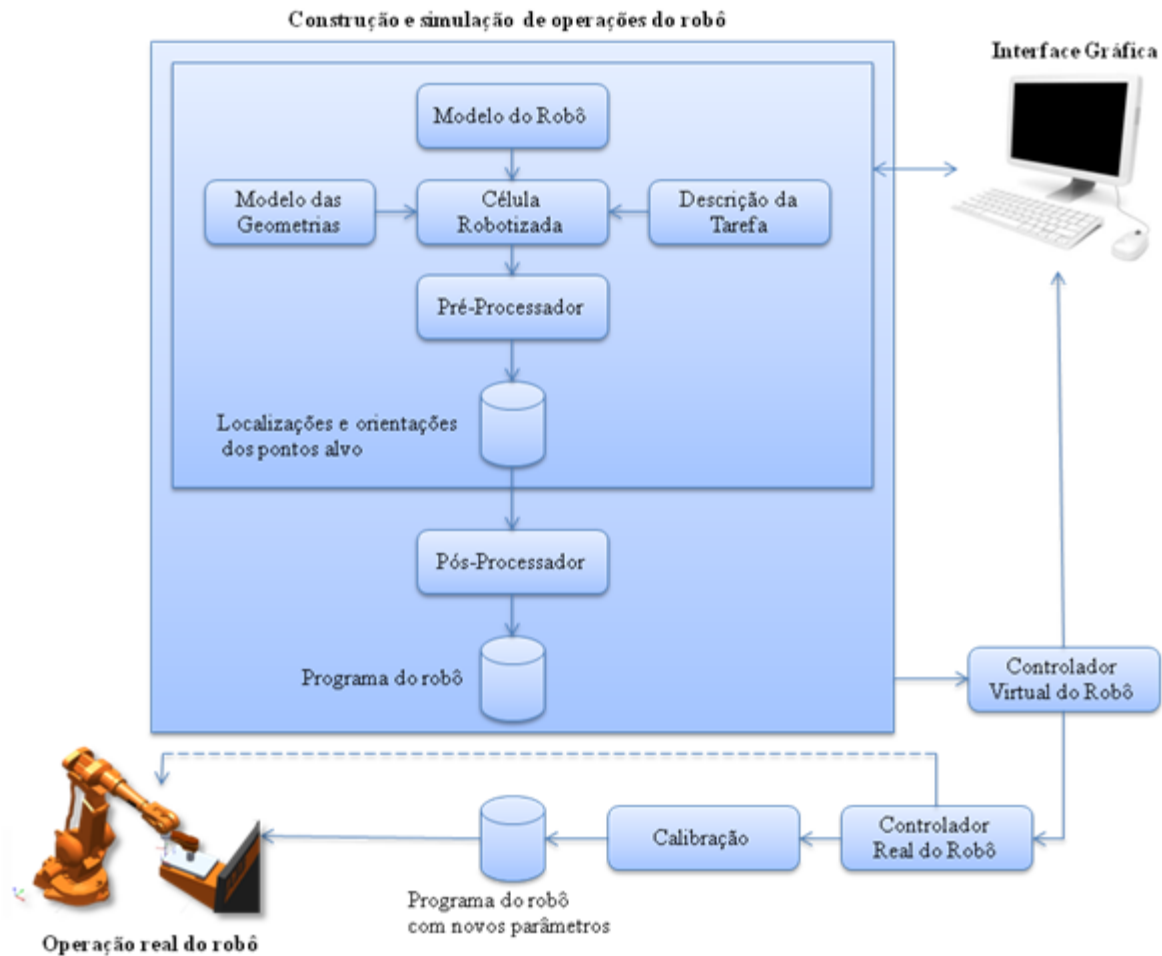
Figura 4.1 - Estrutura de programação *off-line* do *RobotStudio*

A *interface gráfica* do programa proporciona ao programador informações sobre comandos e outras operações necessárias para operar correctamente com o robô. Esta interface, além do propósito acima descrito, também permite a visualização do programa do robô de uma forma tridimensional animada, sendo assim possível a verificação de resultados.

O módulo *modelo do robô* contém informações sobre a forma geométrica, a estrutura cinemática, as relações de dinâmica do robô e de outros elementos controlados pelo controlador do robô (exemplo: mesas de indexação).

O compilar destas informações permite a obtenção de um modelo virtual do robô, baseado no real, possibilitando que se possam observar os movimentos do robô e a verificação de possíveis colisões.

No módulo *modelo das geometrias* estão presentes informações sobre todos os elementos presentes na célula robotizada, desde ferramentas, objectos de trabalho, sistemas de alimentação/remoção, etc.

O módulo *descrição da tarefa* contém todas as informações necessárias para realizar a tarefa pretendida, contendo assim informações referentes às ferramentas, aos referenciais em uso e aos pontos alvo.

No módulo *célula robotizada* encontram-se todas as informações referentes à célula robotizada presente na área de trabalho, uma vez que tem inserida toda a informação vinda dos módulos: modelos de geometrias, modelo do robô e descrição da tarefa.

O *pré-processador* permite criar um conjunto de instruções, quer de movimentação, quer de acção[20], a partir das informações presentes na tarefa de trabalho do robô.

A missão do módulo *pós-processador* é transformar o conjunto de informações presentes no pré-processador e convertê-lo num programa apropriado na linguagem do robô.

Os *controladores* são postos de processamento de dados capazes de interpretar informações inseridas numa linguagem própria. Esta interpretação traduz-se na realização de acções por parte do robô. O controlador virtual representa uma simulação do controlador real, possibilitando a verificação de acções na própria área de trabalho virtual. Caso se verifique uma concordância do modelo virtual para o real, existe a possibilidade de se transferir imediatamente o programa para o controlador real e de o executar sem qualquer calibração.

O módulo *calibração* permite que se procedam a eventuais correcções nos parâmetros do programa, imediatamente antes de o executar, de modo a configurar correctamente o sistema real. Como exemplo, tem-se a redefinição de sistemas de coordenadas de trabalho.

[20] Accionamento de dispositivos através de comandos lógicos

5 Desenvolvimento da Simulação de uma Célula Robotizada

Após efectuado um levantamento acerca dos processos necessários para a concepção de uma célula robotizada são abordados os processos realizados em contexto laboratorial, para a execução da simulação de uma célula robótica com a finalidade de proceder ao acabamento de calçado dotado de solas duplas.

Para efeito de experimentação adoptou-se uma configuração da célula robotizada similar à existente no laboratório de robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto. Esta modelação foi conseguida com o auxílio do *software* de programação *off-line* mencionado anteriormente (*RobotStudio*).

No sentido de permitir a execução do processo de acabamento de sapatos por pessoal não qualificado, idealizou-se, concebeu-se e implementou-se uma solução gráfica para a consola do robô que permite não só facilitar o processo, como também optimizar o mesmo, dada a panóplia de ferramentas de gestão e manutenção presentes neste.

Neste capítulo são abordadas as etapas fundamentais do desenvolvimento dos processos cima descritos.

5.1 Processo de Concepção da Célula Robotizada Virtual

Neste tópico é abordado todo o processo de concepção da célula robotizada virtual a fim de se poder passar à experimentação do processo de acabamento. Esta concepção foi apoiada no esquema apresentado na Figura 4.1 (capítulo quatro), utilizando o *RobotStudio* e o pacote *Machining PowerPac* (*Add-on* do *RobotStudio*, descrito posteriormente).

5.1.1 Instalação e Preparação do *Software* de Programação do Robô

Previamente à modelação da célula robotizada existente no laboratório, foi necessário proceder à instalação de um conjunto de ferramentas informáticas que permite a realização da simulação. Esta foi a sequência de passos seguida na instalação:

- *Microsoft.Net Framework 1.1, 2.0* e *3.5;*
- *ABB RobotStudio 5.11.02* – Base do programa;
- *RobotWare 5.11* – Controlador do Robô;
- *RobotWare Machining 2.10* – Programa de *Machining* do Robô;
- *Machining PowerPac 5.11* – Pacote de interacção da capacidade de *Machining* no *RobotStudio.*

Este *software* foi instalado num computador de secretária, com as seguintes características principais:

- Processador: *Quad Core Q8200* a *2,33 GHz*;
- Memória *RAM*: *4 Gbytes* do tipo *DDR2;*
- Sistema Operativo: *Windows XP Service Pack 3*;
- Placa Gráfica: *Ati Radeon 4830* com *512 Mbytes* de *RAM DD3*.

5.1.2 Obtenção do Controlador Virtual do Robô

É possível criar um controlador virtual no *RobotStudio* recorrendo a uma cópia do controlador real do robô. Para isso efectuou-se uma cópia de segurança do controlador real do robô que foi utilizada para criar um novo projecto no *RobotStudio*. Constatou-se assim que, não só se encontrava ao dispor toda a informação das características reais do robô, como também se tornou imediatamente disponível na área de trabalho do *RobotStudio* a modelação do robô e respectiva mesa (Figura 5.1).
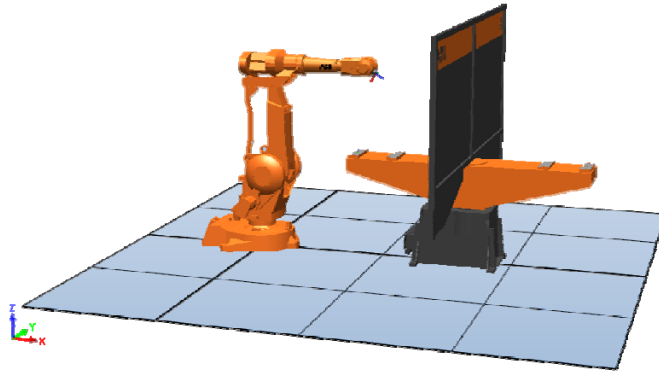
Figura 5.1 – Robô e respectiva mesa, disponíveis na área de trabalho do *RobotStudio*

Assim, dadas estas condições, foi iniciada a construção pormenorizada da célula robotizada real.

5.1.3 Modelação de Geometrias

A modelação das geometrias dos diferentes objectos presentes na célula pode ser concretizada no *RobotStudio*. No entanto, não sendo este *software* específico para modelação tridimensional, apresenta limitações de construção de formas complexas (este apenas apresenta relativa facilidade quando se trata de construir formas simples). Esta limitação foi ultrapassada recorrendo ao uso do *software* de *CAD*, *SolidWorks*, para modelação. As modelações construídas foram guardadas em suportes neutros (como *STEP*, *IGES*, entre outras) e posteriormente importadas pelo *RobotStudio*.

Conjunto sapato e respectivo suporte

Foi construído um modelo tridimensional, que consiste num conjunto representativo de um sapato e respectivo suporte, como mostra a Figura 5.2.
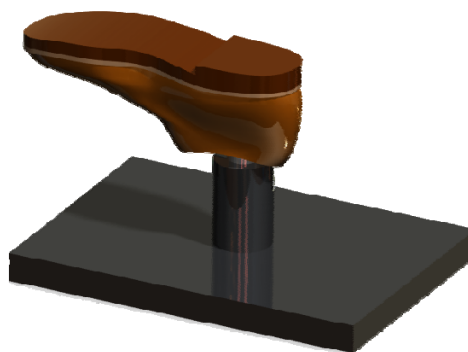
Figura 5.2 - Modelação do conjunto sapato e respectivo suporte

A inserção deste modelo no *RobotStudio* demonstrou alguma fragilidade, quando para a importação do modelo se utilizou o formato *IGES*, uma vez que nem todas as superfícies ficaram delineadas. Assim, voltou-se a importar o modelo no formato *STEP*, mostrando resultados bastante satisfatórios.

Ferramenta

De modo análogo ao considerado no conjunto anterior, foi modelada uma representação tridimensional de uma ferramenta, ilustrada na Figura 5.3.
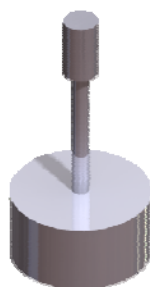
Figura 5.3 - Modelação da ferramenta

Previamente à inserção desta modelação (Figura 5.3) na célula robotizada, foi necessário recorrer a um processo inicial, que permitisse identificar esta modelação como sendo do tipo ferramenta de robô. Assim, em primeiro lugar, foi necessário criar um projecto no *RobotStudio*, exclusivamente para a transformação da modelação proposta em ferramenta, sendo necessário definir um referencial ferramenta e respectivo *Tool Centre Point (TCP)*. Seguidamente, procedeu-se à gravação desta ferramenta (Figura 5.3) na biblioteca de ferramentas, de modo a que, ao ser colocada no projecto, esta seja identificada como ferramenta e automaticamente colocada no elemento terminal do robô. Obteve-se assim a ferramenta com o respectivo *TCP*, como ilustrado na Figura 5.4.
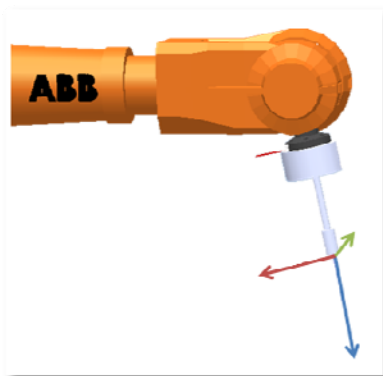
Figura 5.4 - Modelo do Robô com respectiva ferramenta de trabalho

5.1.4 Posicionamento e Descrição da Tarefa

Previamente à descrição da tarefa, é necessário inserir todos os objectos necessários à representação correcta da célula robotizada. Como tal, foi utilizada a biblioteca do *RobotStudio*, que disponibiliza os modelos do gradeamento de protecção e do controlador, de modo a serem inseridos na área de trabalho. Os pilares de detecção de passagem foram modelados utilizando as ferramentas disponibilizadas pelo *RobotStudio*. A fim de se obter um posicionamento correcto de todos os objectos presentes na célula robotizada, foram utilizadas as ferramentas de posicionamento disponíveis no *RobotStudio*. A configuração da célula robotizada com todos os seus elementos constituintes, é apresentada nas Figuras Figura 5.5 e na Figura 5.6.
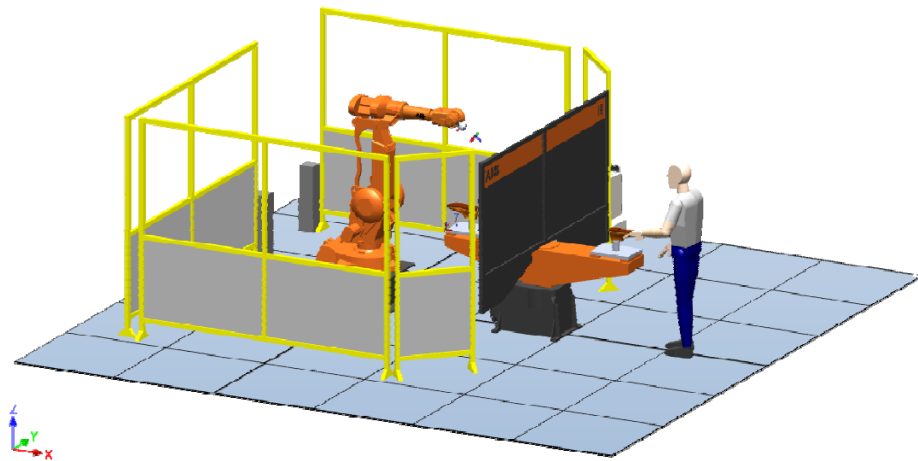
Figura 5.5 - Célula robotizada proposta para simulação da operação de acabamento de calçado
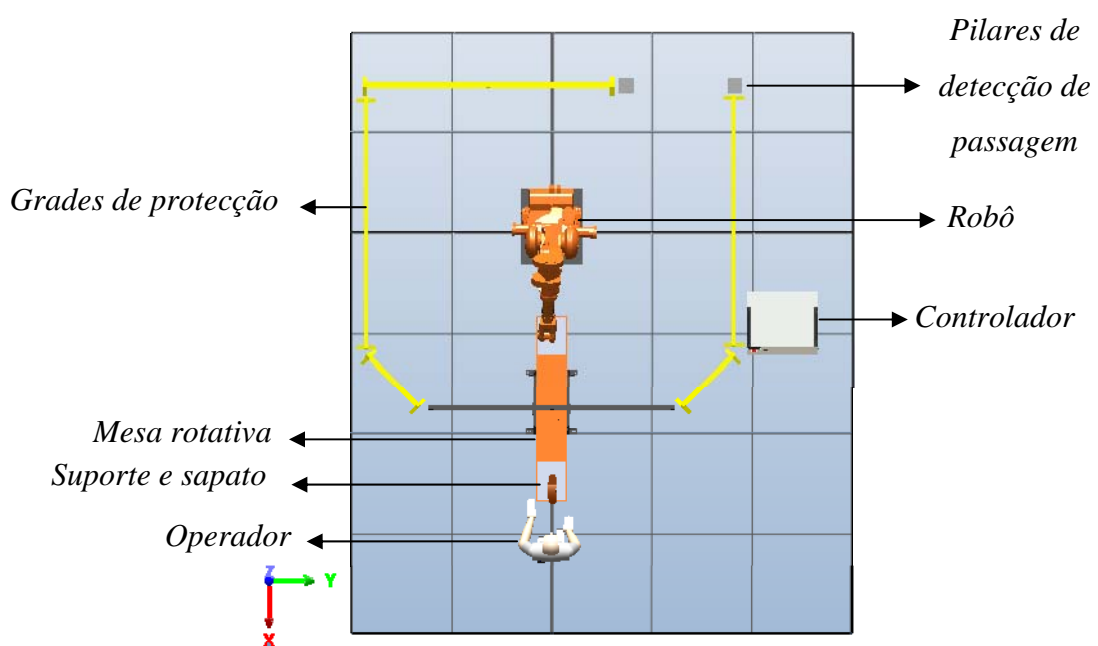
Figura 5.6 - Vista de cima da célula robotizada proposta para simulação da operação de acabamento de calçado

5.1.5 Definição do Referencial de Trabalho

A descrição da tarefa necessita, em primeiro lugar, da colocação apropriada de um referencial de trabalho com o qual todos os pontos da trajectória se encontram relacionados.

O local escolhido foi o ilustrado na Figura 5.7, por ser um ponto pertencente ao local de trabalho e, também, por facilitar a calibração com a célula real (pela calibração de um referencial na mesma localização).
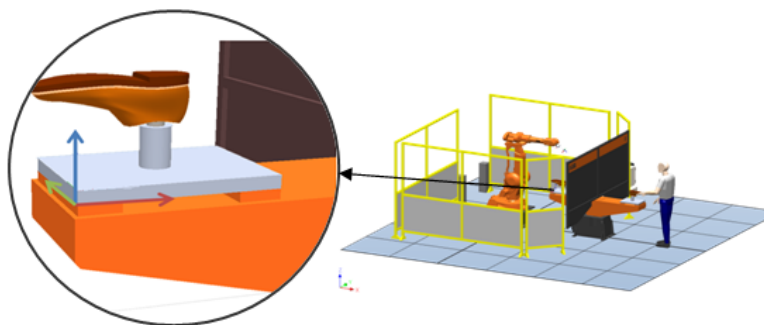
Figura 5.7 - Ilustração da posição do referencial de trabalho

5.1.6 Definição da Trajectória de Trabalho

A definição da trajectória de acabamento do sapato necessita da especificação de um conjunto de pontos e orientações, que o robô utiliza para descrever uma trajectória. Esta trajectória deve assegurar os seguintes requisitos:

- A ferramenta do robô não pode passar a junção da sola superior e do corte do sapato[21] (denominada de zona limite, na Figura 5.8), caso contrário arrisca a danificar o sapato;
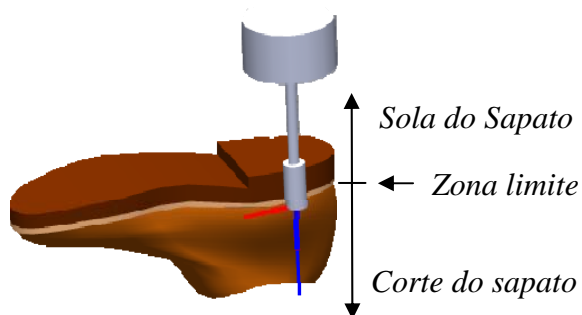
Figura 5.8 - Definição da zona limite entre as duas partes principais do sapato

[21] Denomina-se de corte do sapato a forma que se acomoda à parte superior do pé, normalmente feita de pele.

- A direcção de actuação da força de contacto da ferramenta deve ser mantida perpendicular à superfície da sola, de modo a ser possível garantir um acabamento uniforme (ver Figura 5.9);
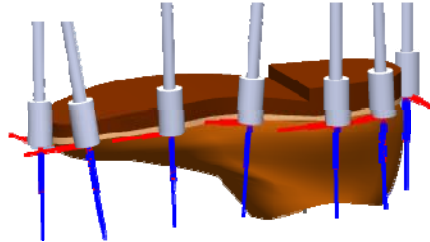
Figura 5.9 - Ferramenta a actuar segundo a direcção perpendicular à superfície da sola do sapato[22]

5.2 Processo de Simulação da Célula Robotizada

O modo tradicional de programar um robô consiste em definir uma evolução espacial e temporal do robô, conhecida por trajectória. Uma vez definida, os parâmetros da trajectória são mantidos constantes, independentemente das forças envolvidas no processo, mesmo que a trajectória definida não coincida com a superfície em que se deseja operar. Tal facto poderá ter consequências relativamente à qualidade do produto, no caso de este, por exemplo, não estar colocado devidamente, bem como o risco de danificação de equipamentos.

De forma a poder ultrapassar estes problemas, a *ABB* desenvolveu uma solução de comando de robôs que envolve o uso de controlo de força. Esta implementação requer o uso de uma aplicação informática denominada de *RobotWare Machining Force Control*, que permite que o robô possa manter a ferramenta em contacto com as superfícies de trabalho, independentemente da trajectória programada.

[22] Sendo a direcção perpendicular obtida pela regra da mão direita, em que, a cor azul do referencial representa o eixo dos *ZZ* e a vermelha o eixo dos *XX*.

5.2.1 Aplicação do *Machining PowerPac*

Efectuado o estudo sobre a definição da trajectória, procedeu-se ao uso da aplicação *Machining PowerPac*, de modo a poder testar-se a aplicabilidade do uso do robô para acabamentos de calçado, apoiada sobre a tecnologia de controlo de força. A aplicação dispõe de dois modos de operação: o *FC Pressure* e o *FC SpeedChange*.

O *FC Pressure* tem como objectivo tornar o robô sensível a forças de contacto, permitindo que este se possa adaptar às superfícies de trabalho. Para concretização deste objectivo, é necessário ensinar ao robô uma trajectória de referência, definindo a direcção sobre a qual o robô aplicará uma dada força. Assim, o robô poderá modificar cada ponto da sua trajectória de referência, de modo a satisfazer a força especificada. De modo a compreender melhor esta função, poderá observar-se o exemplo ilustrado na Figura 5.10.
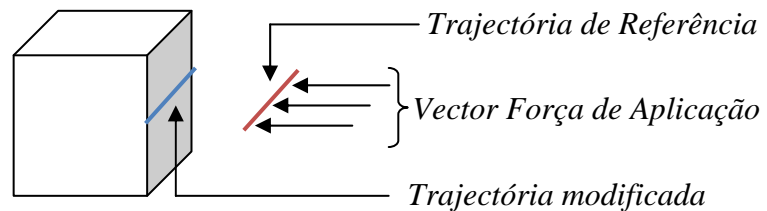
Figura 5.10 – Exemplo de aplicação do *FC Pressure*

O *FC SpeedChange* tem como objectivo controlar a velocidade da trajectória a partir de uma força de contacto limite. Desta forma, quando se excede um valor de força limite, a velocidade irá ser automaticamente reduzida ou, contrariamente, aumentada caso haja um decremento da força. Contrariamente ao *FC Pressure*, neste tipo de operação o percurso definido irá ser cumprido. Na Figura 5.11 encontra-se ilustrado um exemplo desta função.
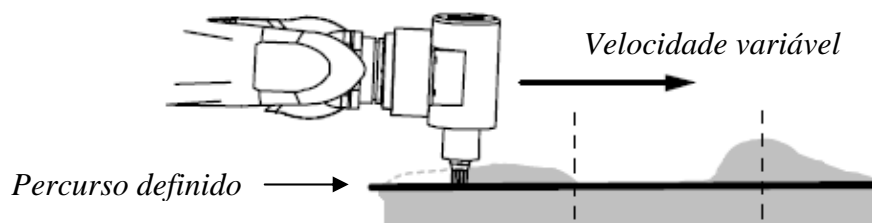
Figura 5.11 – Exemplo de aplicação do *FC SpeedChange*[23]

[23] Figura retirada do manual *Force Control for Machining* (ABB, 2007)

No que respeita à obtenção da simulação virtual, de facto, não é relevante a escolha objectiva do tipo de operação, uma vez que os resultados visuais serão iguais (o *software* não simula as forças de interacção).

A obtenção de uma solução baseada no *Machining PowerPac*, requer a utilização de uma metodologia apropriada. Esta encontra-se representada no esquema da Figura 5.12.
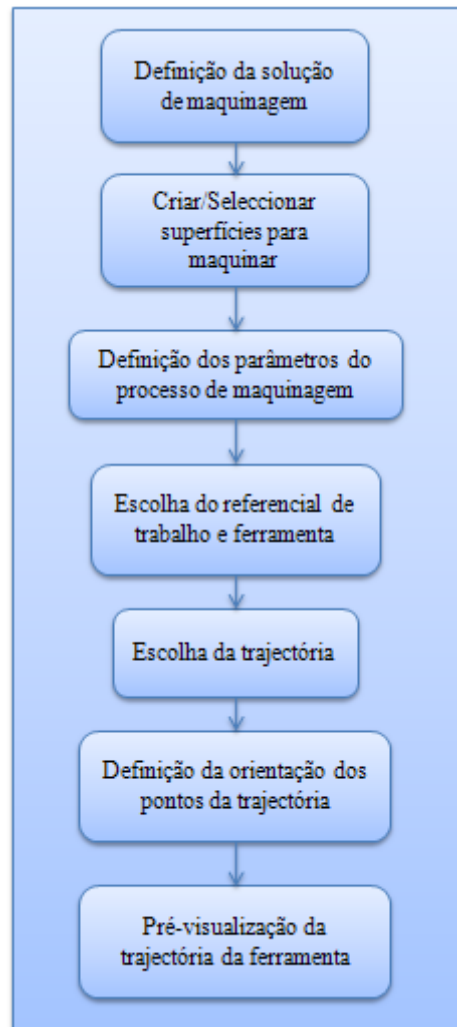
Figura 5.12 - Metodologia utilizada para implementação da solução[24]

5.2.2 Obtenção da Solução

Considerando a metodologia proposta e adequando-a ao propósito deste trabalho que, como referido, visa a obtenção de uma trajectória que permita realizar o acabamento de solas

[24] Apoiado no manual de programação do *Machining PowerPac* (ABB, 2008)

de sapato, são de seguida apresentados por tópicos os principais passos, de modo a simplificar a compreensão do processo.

- ***1º - Definição da solução de maquinagem*** **(**Figura 5.13**)**
    - Escolha do controlador do robô a efectuar o acabamento;
    - Escolha da tarefa em que se encontra o robô;
    - Escolha do nome para a solução de acabamento.
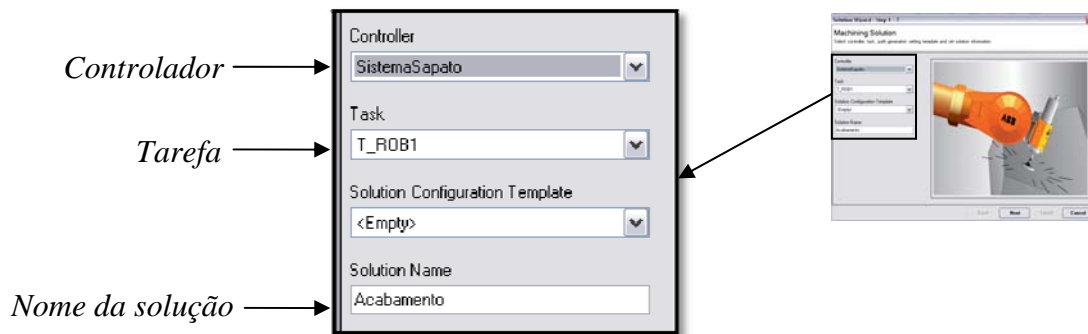
Figura 5.13 - Escolha do controlador, da tarefa e do nome da solução

- ***2º - Criar/Seleccionar superfícies para maquinar*** **(**Figura 5.14**)**
    - Selecção de toda a parte lateral da sola do sapato.
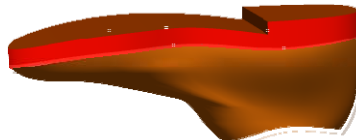
Figura 5.14 - Selecção da superfície lateral da sola do sapato (a vermelho)

- ***3º - Definição dos parâmetros do processo de maquinagem*** **(**Figura 5.15**)**
    - Escolha do tipo de processo de maquinagem (*FC Pressure*);
    - Escolha da direcção do eixo dos *YY* para aplicação da força;
    - Escolha do valor da força durante as diferentes fases do processo (encosto, durante processo, afastamento);
    - Escolha de outros parâmetros do processo (deixados em modo padrão por não serem cruciais à simulação).
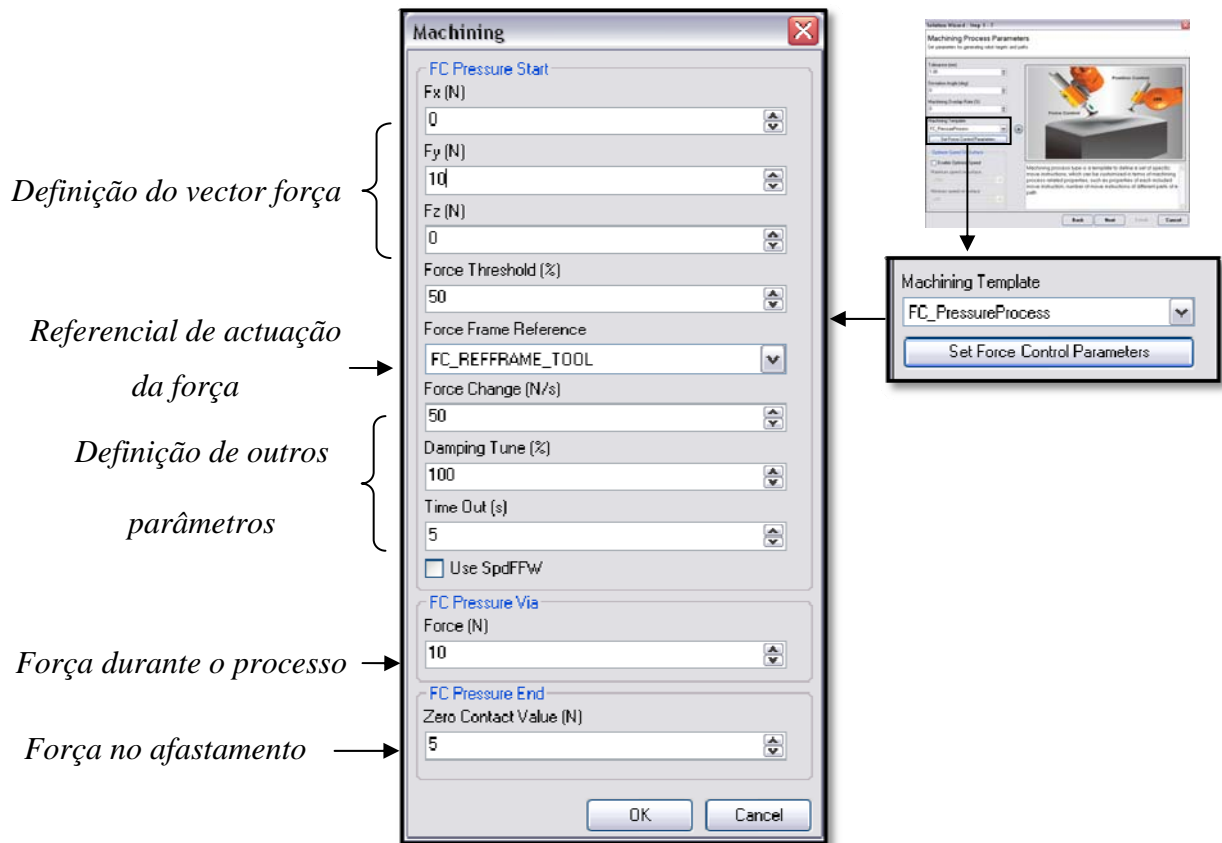
Figura 5.15 - Definição dos parâmetros do processo de maquinagem

- ***4º - Escolha do referencial de trabalho e da ferramenta*** (Figura 5.16)
    - o Activação do referencial de trabalho definido previamente;
    - o Activação da ferramenta e definição das suas características.
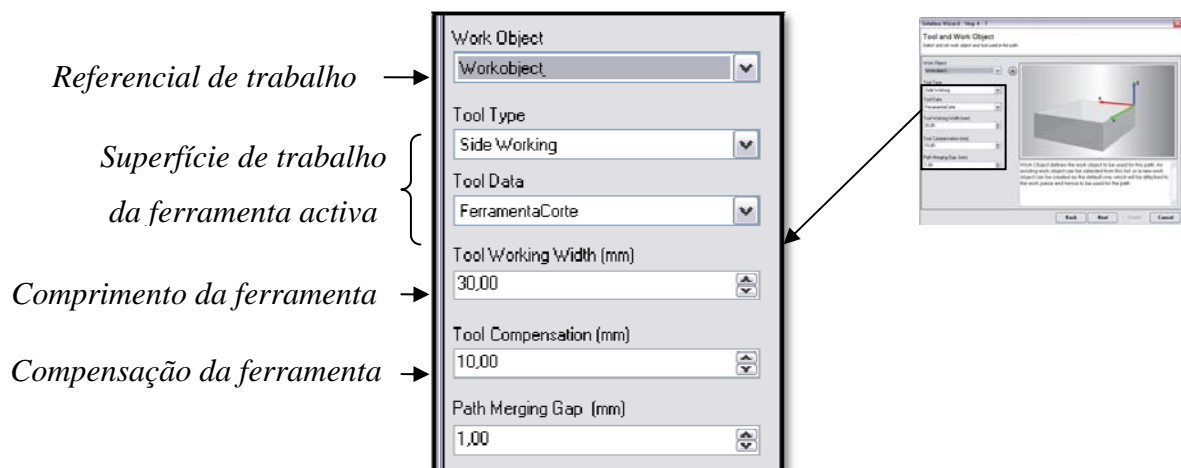
Figura 5.16 - Activação do referencial de trabalho e características da ferramenta

- ***5º - Escolha da trajectória* (**Figura 5.17**)**
    - Selecção da junção entre a sola superior e o corte do sapato.
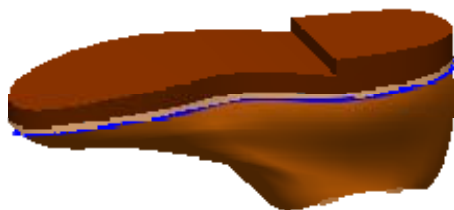
Figura 5.17 - Selecção da trajectória (a azul)

- ***6º - Definição da orientação dos pontos da trajectória* (**Figura 5.18**)**
    - Escolha da orientação perpendicular à superfície;
    - Escolha do tipo de trajectória de encosto e de afastamento.
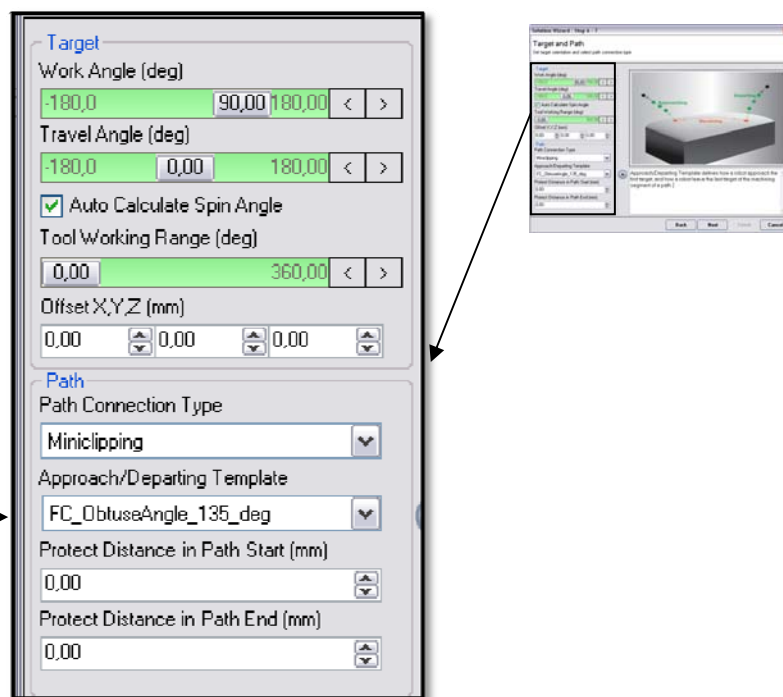
Figura 5.18 - Definição dos parâmetros dos pontos da trajectória

- ***7º - Pré-visualização da trajectória da ferramenta*** (Figura 5.19)
    - Verificação da trajectória e respectivo sentido a efectuar pela ferramenta;
    - Determinação da configuração do robô.
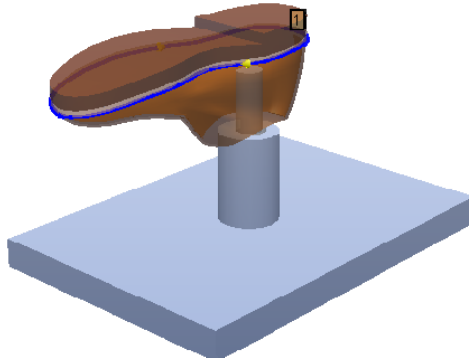
Figura 5.19 - Pré-visualização da trajectória (a azul)

5.2.3 Simulação da Solução

Posteriormente à pré-visualização da trajectória, foi criada a solução, na qual, a orientação dos referenciais associados aos pontos que definem a trajectória se encontra ilustrada na Figura 5.20. Esta encontra-se legendada da seguinte forma:

- Direcção do eixo dos *XX* – cor vermelha (1);
- Direcção do eixo dos *YY* – cor verde (2);
- Direcção do eixo dos *ZZ* – cor azul (3);
- Superfície de encosto sobre a qual actua a força em *YY* – cor amarela (4).
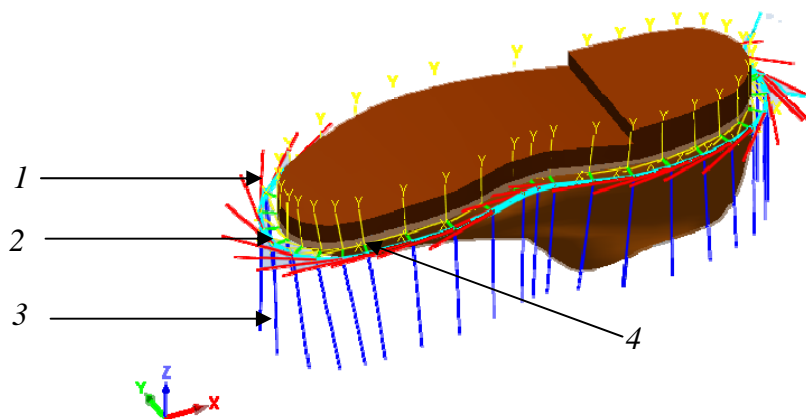
Figura 5.20 - Orientação dos pontos pertencentes à trajectória

A trajectória de acabamento tem como movimentação o sentido dos ponteiros do relógio (Figura 5.21). A razão que levou à atribuição deste sentido foi meramente aleatória, uma vez que na simulação, a ferramenta não tem qualquer movimento de rotação, o que faz com que seja indiferente a escolha do sentido da movimentação.
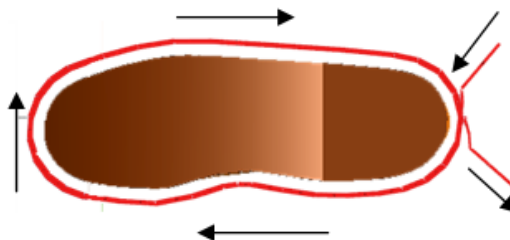
Figura 5.21 - Sentido horário da trajectória de acabamento

A simulação da solução requereu que se fizesse previamente o sincronismo da solução proposta para o controlador virtual do robô. Desta forma, tornou-se possível simular o processo de acabamento, usufruindo das características cinemáticas do robô real. Esta simulação encontra-se ilustrada na Figura 5.22, durante o decorrer do acabamento.
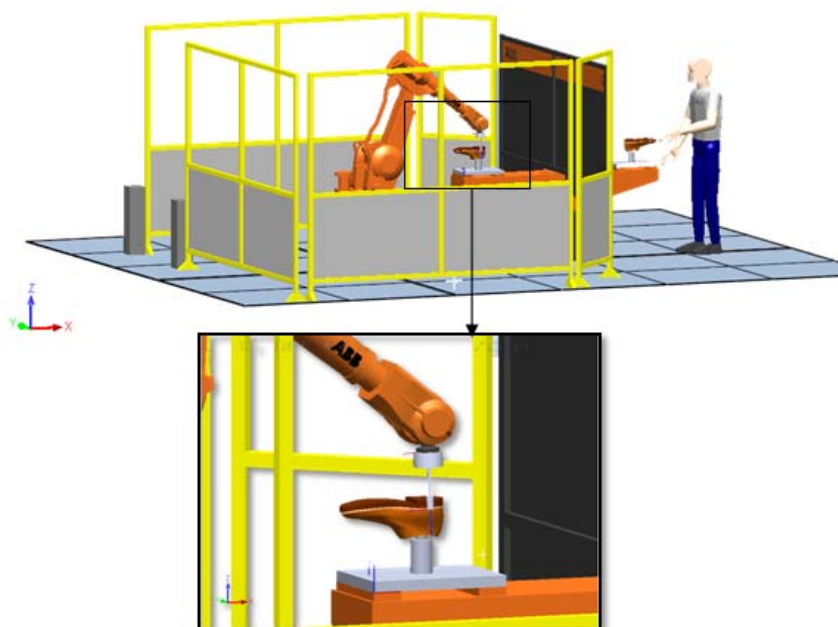
Figura 5.22 - Ilustração da simulação da operação de acabamento de sapatos

O facto de se ter feito o sincronismo para o controlador virtual, possibilitou obter a programação deste processo na linguagem do robô, *RAPID*. Na ilustração seguinte (Figura 5.23), são mostrados alguns pormenores sobre esta programação.

Figura 5.23 – Pormenores do algoritmo de realizar acabamentos em calçado

5.3 Desenvolvimento de uma Aplicação Gráfica para Operação da Célula Robotizada

A necessidade de conceber uma aplicação gráfica para a consola do robô surgiu com o intuito de permitir que qualquer encarregado de operações (não necessariamente especializado em robótica) seja capaz de operar a célula robotizada. O robô deve poder efectuar acabamentos em qualquer sapato, uma vez que todos os sapatos se encontrarão disponíveis numa base de dados construída para o efeito. O manual de utilização desta aplicação encontra-se apresentado no Anexo A – Manual de utilização da aplicação desenvolvida.

Esta solução gráfica deve apresentar como principais características:

- Ter a possibilidade de ser operada de modo visualmente intuitivo por pessoal não especializado;
- Apresentar soluções de recurso em caso de falha em operação;
- Permitir a utilização de poucos recursos de memória;
- Incorporar uma base de dados com os diferentes modelos e tamanhos dos sapatos a processar;
- Ter flexibilidade para poder ser utilizado em configurações da célula robotizada quer com o robô a movimentar a ferramenta, quer com o robô a movimentar o sapato.

A concepção desta aplicação foi baseada na linguagem *C Sharp*, recorrendo à ferramenta de programação o *Microsoft Visual Studio 2005*, sobre a qual se teve de instalar o *software RobotStudio Application Builder 5.10*. O uso desta aplicação permitiu aceder a uma base de dados de programas concebidos especialmente para facilitar na construção de programas destinados à consola do robô (denominados no termo inglês de *namespaces)* (Sharp, 2008). O código integral de programação da aplicação é disponibilizado e pode ser consultado no Anexo B – Código da Aplicação Desenvolvida.

5.3.1 Estrutura de Funcionamento do Menu Principal

Antes de passar à programação propriamente dita, foi necessário pensar sobre qual teria de ser o primeiro menu, de modo a facilitar uma futura abordagem. Com isto surgiu o esboço apresentado na Figura 5.24.
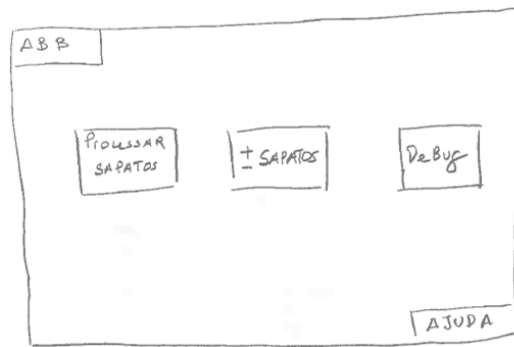
Figura 5.24 – Esboço inicial do menu principal do programa

Este esboço (Figura 5.24) revelou-se como um ponto importante na concepção da aplicação, por terem sido idealizados os seus futuros submenus, representados por "botões" no esboço da Figura 5.24.

Posteriormente a esta primeira abordagem, iniciou-se a fase de desenvolvimento da estrutura de funcionamento, baseada na concepção visual definida. Assim, o menu principal engloba as três funções principais esquematizadas na Figura 5.25.
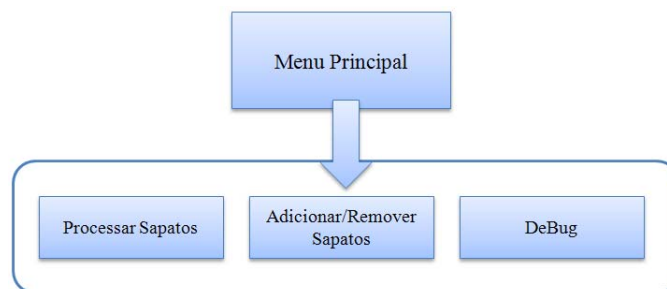
Figura 5.25 - Funções do Menu Principal

As funções a implementar nos botões (que permitem aceder a novos menus quando activados), possibilitam a execução das seguintes tarefas:

- *Processar Sapatos* – permitir a selecção do sapato a trabalhar e iniciar processamento;
- *Adicionar/Remover Sapatos* – disponibilizar a possibilidade de adicionar ou remover sapatos da base de dados existente;

- *DeBug* – permitir a paragem da execução do programa, a desactivação do controlo de força e mover o robô para uma posição de referência.

5.3.2 Estrutura de Funcionamento do Submenu *Processar Sapatos*

Na estrutura de funcionamento do submenu *Processar Sapatos*, que tem como finalidade a execução de uma trajectória de corte, é necessário garantir que alguém responsável tenha à sua disposição:

- A possibilidade de escolher um modelo de sapato com o respectivo número e orientação (pé esquerdo/direito);
- Rodar a mesa e respectivo suporte do sapato acoplado;
- Paragem de emergência do robô.

1ª Versão da estrutura de funcionamento

A primeira versão da estrutura de funcionamento desenvolvida consistia basicamente num “caminhar” de forma lógica e sequencial sobre vários menus, permitindo ao utilizador a escolha do modelo, número e orientação do sapato, onde, posteriormente este teria acesso a uma ferramenta que permitisse executar o programa na janela *Execução*. Esta versão pode ser vista na Figura 5.26
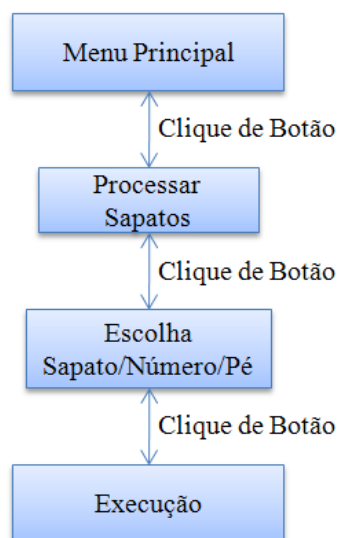
Figura 5.26 – Primeira versão do submenu *Processar Sapatos*

Apesar de ter sido uma primeira versão, simples, levantou de imediato alguns constrangimentos que se revelaram bastante pertinentes.

Verificou-se que é possível passar informação de um menu para outro, no entanto esta funcionalidade pode acarretar sobrecargas de memória do controlador do robô. Isto acontece porque, por exemplo, quando se torna necessário utilizar informações vindas do menu *Escolha Sapato/Número/Pé*, este terá de permanecer aberto até que um outro menu (por exemplo o menu *Execução*) retire a informação que necessita. Esta necessidade, faz com que seja preciso ter sempre tantos menus abertos quanto os menus que disponibilizam a informação requerida. Esta multiplicidade de menus provoca um aumento da carga computacional do controlador, provocando atrasos de processamento, quebras visuais momentâneas no ambiente gráfico e eventualmente erros inesperados.

2ª Versão da estrutura de funcionamento

A fim de evitar problemas associados ao excesso de sobrecarga de memória, é indispensável que, ao passar de um menu para um outro, se feche e limpe da memória o respectivo conteúdo (ABB, 2006). Este procedimento condiciona o processo de preservar informações que são necessárias de ser transferidas de um menu para outro. Uma possível solução passa por englobar num único menu todas as informações a processar. Isto levou à adopção da utilização de *tabs*[25] (Figura 5.27), como forma de permitir, não só ter vários pseudo-menus, como também aceder a informações vindas de uma qualquer *tab* do mesmo menu.
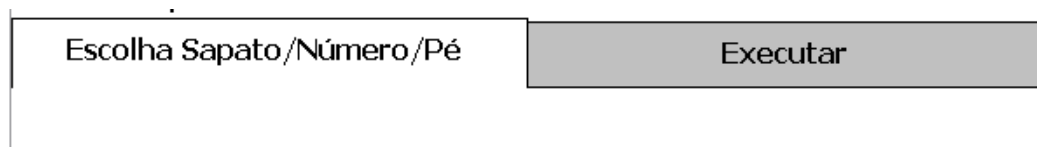
Figura 5.27 – Exemplificação do que são *tabs* (duas *tabs*, nesta ilustração)

Foi, assim, definida uma nova estrutura de funcionamento, apresentada na Figura 5.28, na qual se adicionou um novo *tab* com a função de garantir que o utilizador tenha alguns alertas que o ajudem essencialmente em caso de enganos, distracções e excessos de confiança.

---

[25] A tradução de *Tab* para português significa etiqueta ou secção.
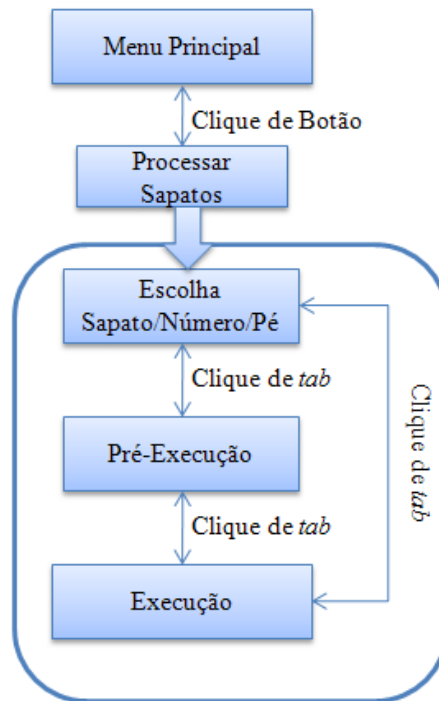
Figura 5.28 - Segunda versão do submenu *Processar Sapatos*

A estrutura apresentada na Figura 5.28 foi implementada e testada, tendo revelado algumas limitações:

- O utilizador pode mudar de *tab* "saltando" de uma janela activa para uma outra qualquer, o que pode comprometer a correcta introdução de dados ou mesmo o esquecimento de introdução de dados necessários;
- Ausência de confirmação das selecções das operações feitas pelo operador.

Versão final da estrutura de funcionamento

Estes testes levaram ao desenvolvimento de uma nova versão da estrutura de funcionamento, apresentada na Figura 5.29.
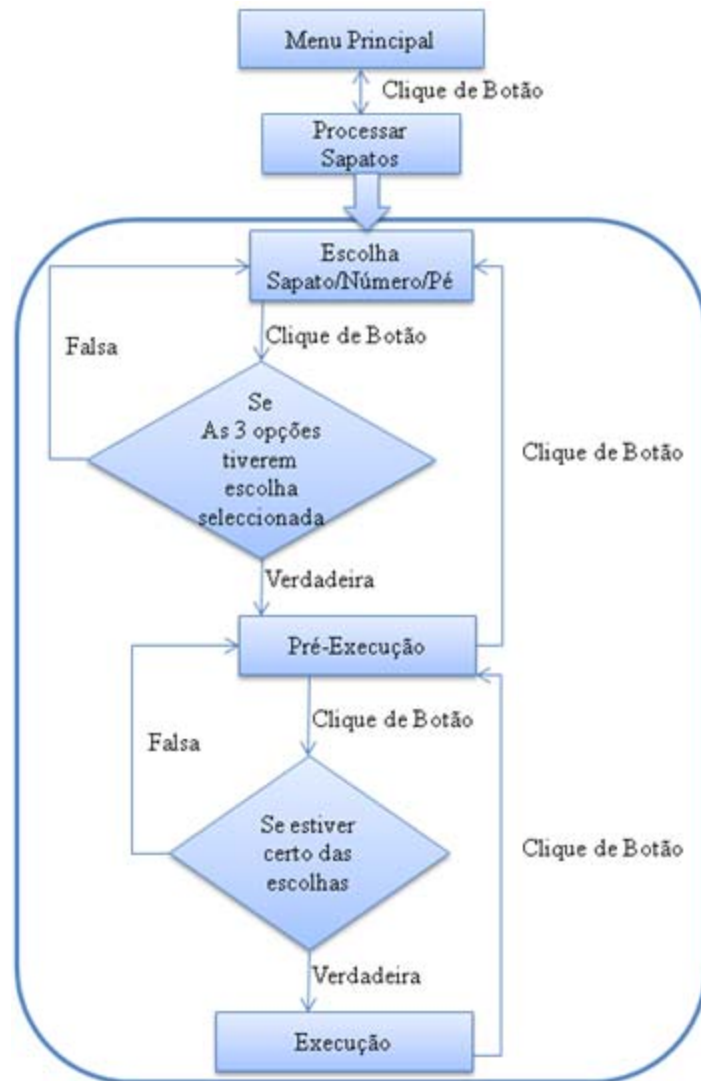
Figura 5.29 - Versão final do submenu *Processar Sapatos*

Esta solução garante que o operador tenha obrigatoriamente de passar por todos os passos. Desta forma, é-lhe oferecida alguma segurança relativamente ao que está a fazer. Aliado a este facto, foram também introduzidas mensagens de texto de confirmação e validação (*dialog box*) para que seja possível reconfirmar as escolhas feitas.

A solução encontrada envolveu a escrita de código que inibe a activação das *tabs*, sendo a comutação entre as *tabs* substituída por recurso a botões de navegação (botões de “próximo” e “anterior”), que permitem assim aceder, de forma ordenada, aos conteúdos associados a cada *tab*. Esta estrutura de funcionamento adoptada para este bloqueio, encontra-se ilustrada na Figura 5.30.
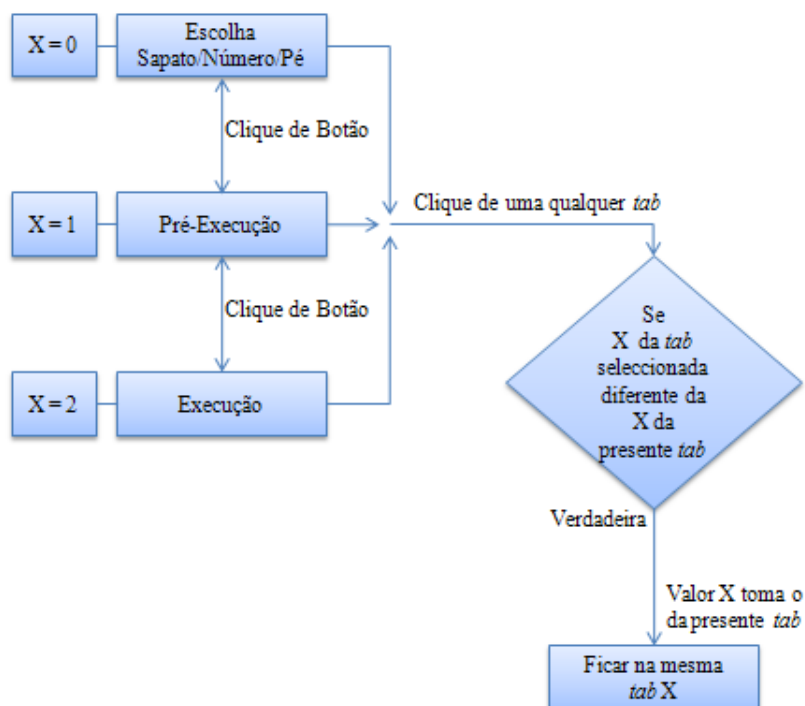
Figura 5.30 – Estrutura de funcionamento do bloqueio das *tabs*

5.3.2.1 Escolha Sapato/Número/Pé

A *tab Escolha Sapato/Número/Pé* apresenta três listas que permitem ao operador seleccionar as características do sapato a processar, ou seja, deve permitir trabalhar com diferentes modelos, tamanhos e orientação do pé. Este requisito requer que exista uma base de dados com toda esta informação. Surgiram então duas opções possíveis: uma base de dados independente da aplicação ou uma base de dados incorporada na aplicação.

A escolha recaiu sobre uma base de dados independente por permitir que seja possível a um utilizador proceder à escolha de um sapato de uma lista, e quando o fizer, seja actualizada automaticamente a lista dos números correspondentes ao modelo escolhido. Foi importante realizar esta observação, uma vez que um dado modelo não tem necessariamente de ter a mesma gama de números de um outro. Isto permite que o operador não tenha de recorrer a qualquer lista externa à consola ou até mesmo saber de cor os números que cada modelo possui. A orientação do pé irá permanecer sempre constante, como tal, não necessita de vir de uma base de dados, mas sim da concepção da aplicação (informação vinda do design do programa). Outra vantagem foi a de possibilitar o adicionar e remover sapatos, como inicialmente previsto na metodologia da aplicação (Figura 5.24), funcionalidade que a opção

de uma base de dados incorporada não permitia (dado que toda a informação seria vinda da concepção da aplicação).

A opção com base de dados independente desenvolvida encontra-se ilustrada na Figura 5.31.
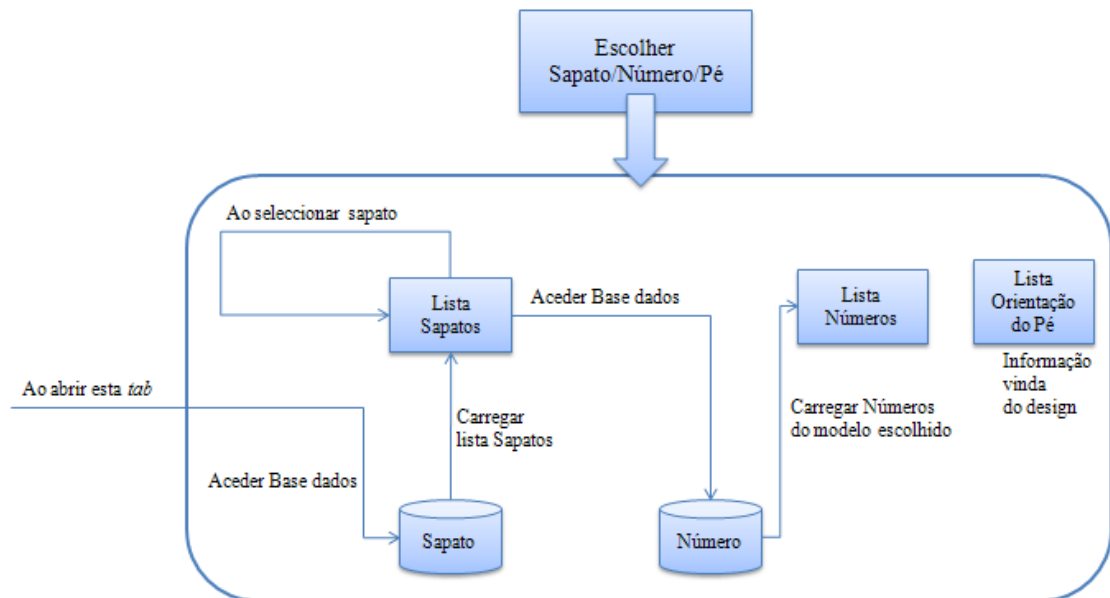
Figura 5.31 – Estrutura de funcionamento da *tab Escolher Sapato/Número/Pé*

5.3.2.2 Pré-Execução

A *Pré-Execução* tem como principal objectivo fornecer ao operador uma visualização das escolhas efectuadas, garantindo que este está certo quanto às opções que tomou.
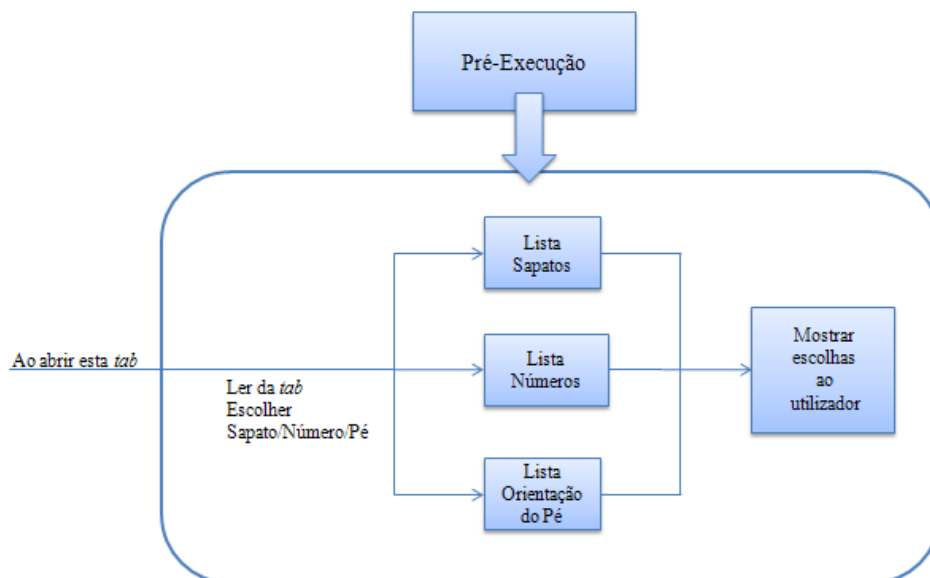
Figura 5.32 – Estrutura de funcionamento da *tab Pré-Execução*

A Figura 5.32 ilustra o algoritmo desenvolvido, onde ao abrir a presente *tab* possam ser lidos os valores que foram seleccionados anteriormente (*tab Escolher Sapato/Número/Pé*) e apresentados, nesta nova *tab*, de um modo visualmente adequado. Esta operação só se torna possível de realizar por ter sido adoptada a estratégia de utilização de *tabs* numa mesma janela (interligações de informação).

5.3.2.3 *Execução*

Após terem sido efectuadas as escolhas sobre o sapato e suas características, é necessário oferecer ao utilizador ferramentas que lhe permitam iniciar o acabamento do respectivo sapato.

A solução concebida para esta operação foi estruturada conforme se apresenta na Figura 5.33.
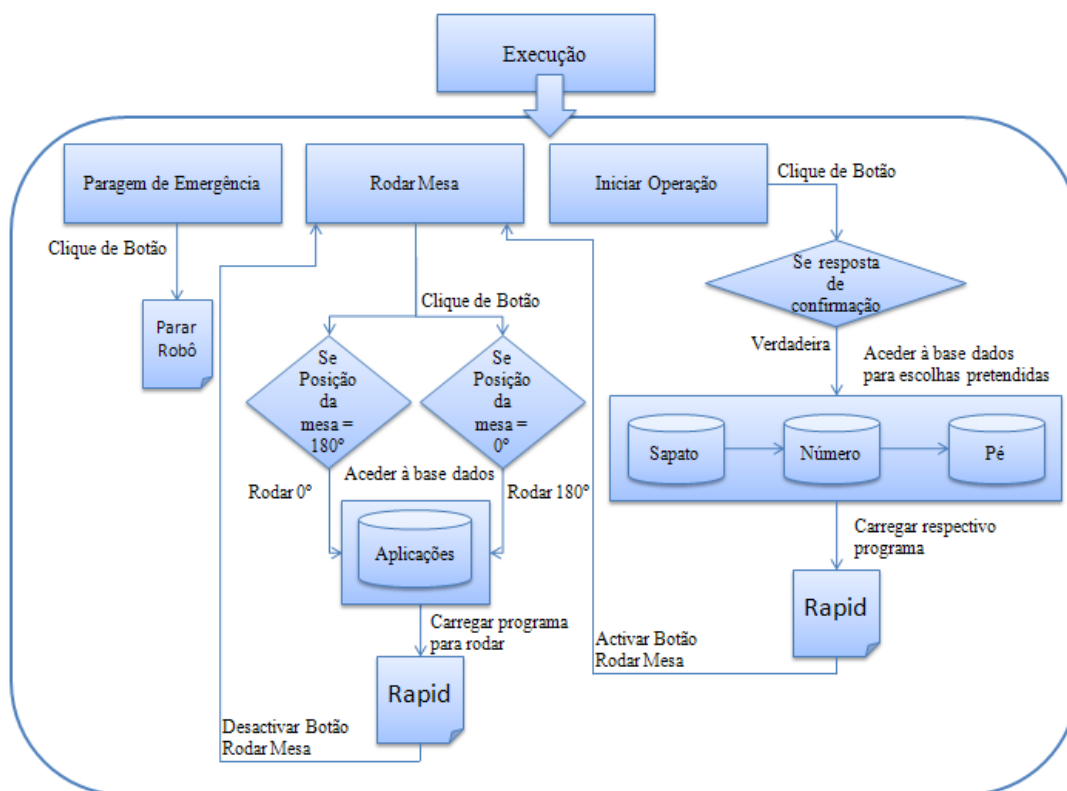
Figura 5.33 – Estrutura de funcionamento da *tab* Execução

Nesta fase, o operador está no limiar de poder executar o acabamento no sapato pretendido. Para dar ordem de execução, o operador apenas terá de carregar no botão “Iniciar Operação”, permitindo que a aplicação possa assim proceder ao carregamento do programa de acabamento (escrito em *RAPID*), para a escolha efectuada previamente. Posteriormente,

quando o acabamento termina, activar-se-á o botão "Rodar Mesa" permitindo que o operador rode a mesa e volte a poder processar um outro sapato. Em caso de emergência, o utilizador tem ao seu dispor o botão "Paragem de Emergência" que, em qualquer situação, ao ser premido, imediatamente executará uma paragem sem qualquer questão de confirmação.

5.3.3 Estrutura de Funcionamento do Submenu *Adicionar/Remover Sapatos*

O acesso à base de dados é feito através da consola de programação do robô. Para esse efeito foi desenvolvido o código necessário que permite adicionar ou remover informação relativa aos sapatos. De outra forma, esta informação teria de ser transferida para um computador, onde sofreria modificações, para posterior reenvio para o controlador. As desvantagens associadas a esta consideração traduzem-se, quer em termos de perdas de tempos, quer em possíveis perdas de informação por modificação descuidada. Outras vantagens associadas são obviamente o fácil acesso à base de dados, podendo-se adicionar ou remover modelos de sapatos. Um factor importante a considerar foi condicionar o acesso a esta funcionalidade apenas a pessoal que tenha permissões, através do conhecimento de uma palavra-chave. A ilustração seguinte, Figura 5.34, descreve todo este processo, acima enunciado.
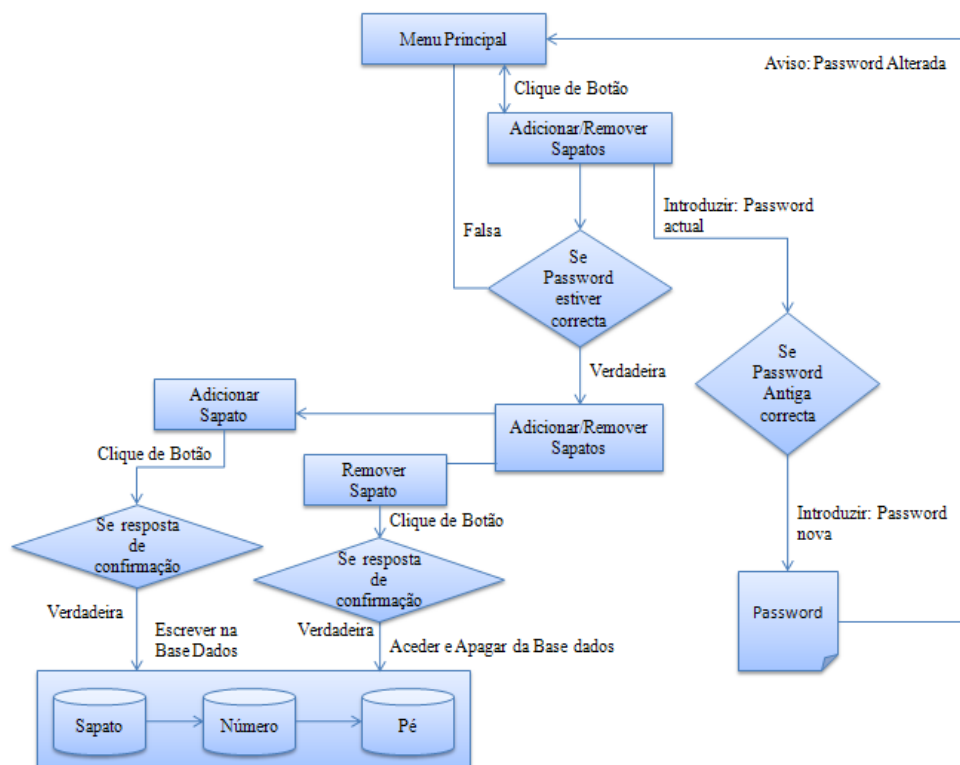
Figura 5.34 – Estrutura de funcionamento de Adicionar/Remover Programas

Como exemplo prático do uso desta funcionalidade, pode descrever-se a situação de um determinado modelo deixar de ser processado por um determinado período de tempo. Poder-se-á retirar o mesmo da base de dados, permitindo não só encurtar a lista de sapatos, como garantir uma menor margem de engano por parte do utilizador na escolha do modelo para processar. Caso este modelo volte a ser necessário, poderá ser reposto rapidamente, bastando para isso adicionar novamente na base de dados o seu nome.

5.3.4 Estrutura de Funcionamento do Submenu *DeBug*

Dada a necessidade de recuperar de possíveis falhas que possam ocorrer durante a execução dos programas de acabamento, foi necessário desenvolver a estrutura de funcionamento ilustrada na Figura 5.35.
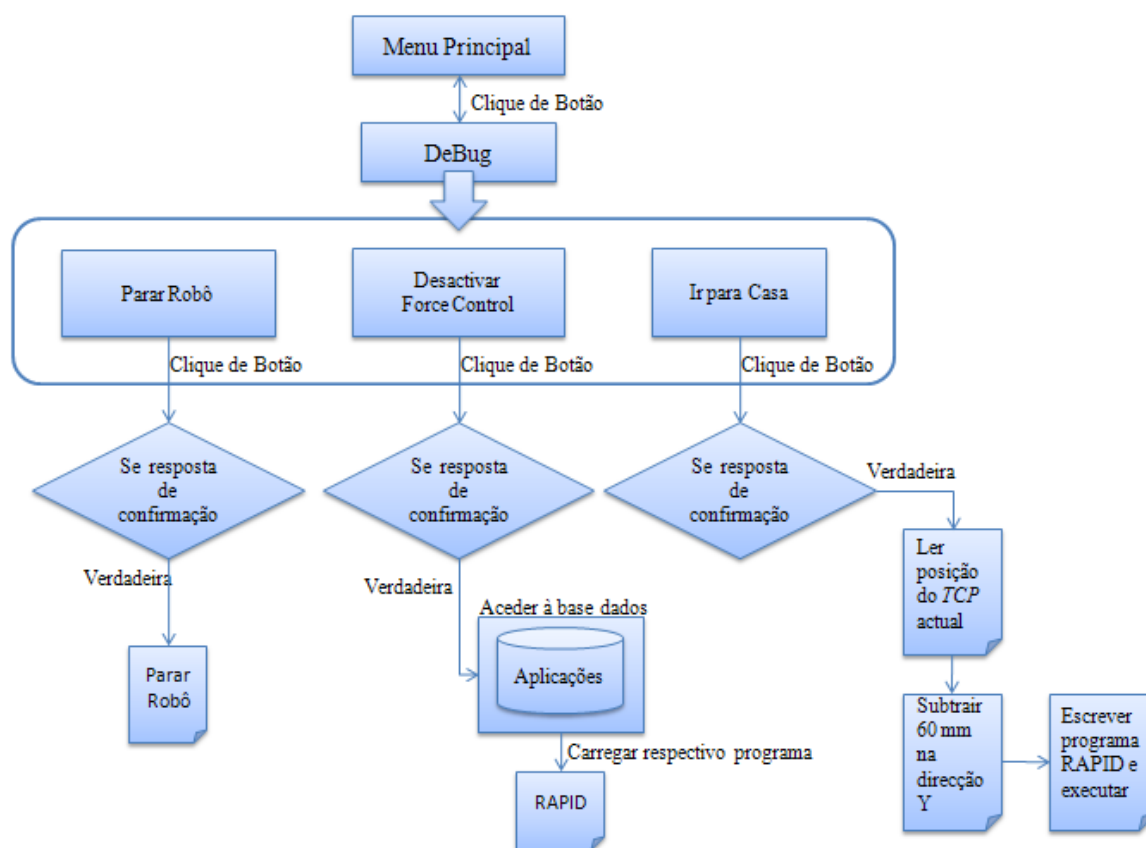
Figura 5.35 – Estrutura de funcionamento *DeBug*

A escolha das opções presentes nesta estrutura, foi apoiada sobre experiências efectuadas utilizando o controlo de força, no qual se verificou que, caso sucedam anomalias no decorrer do programa, como a não correspondência visual da operação desejada, ter-se-á de:

- Parar o robô (no programa será questionado ao utilizador sobre a sua certeza, uma vez que esta não terá de ser necessariamente de emergência);
- Desactivar o controlo de força antes de efectuar qualquer operação de manutenção (como recorrer ao *jogging* do robô), uma vez que podem ocorrer forças de reacção (manifestadas como oscilações não controladas no robô);
- Ir para uma posição de referência (casa), em que seja garantido que o robô se desloque primeiramente na direcção contrária e perpendicular à superfície, de forma a afastar-se, garantindo que não colide com o sapato; posteriormente, a movimentação até à posição casa pode ser cumprida com sucesso (ilustração desta sequência na Figura 5.36).
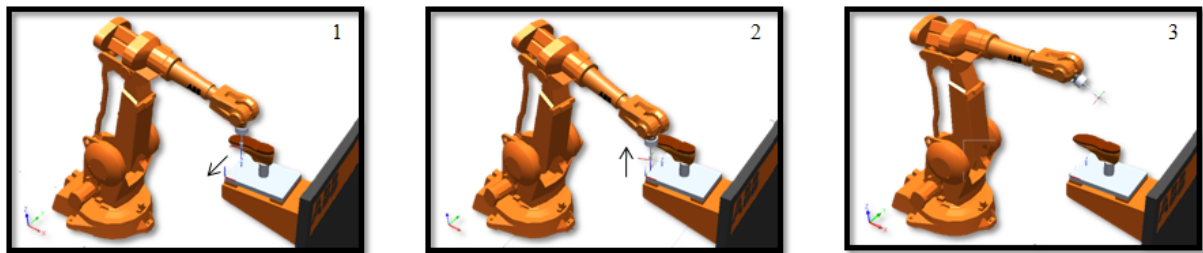
Figura 5.36 – Sequência de movimentação referente à ordem “Ir para Casa”

6 Implementação da Solução

Verificadas as condições de execução do acabamento de solas duplas de sapato em ambiente virtual, tomou-se a iniciativa de efectuar testes laboratoriais, operando na célula robotizada real.

Este processo surgiu na óptica de se obter uma maior sensibilidade acerca da aplicabilidade do uso de controlo de força na implementação de soluções em calçado real. Este facto deveu-se à inexistência de um *feedback* das forças envolvidas no processo por parte do uso do *software* de programação *off-line*, no qual apenas houve uma visualização cinemática de todo o processo. Esta implementação passou pelas seguintes fases:

- Fixação apropriada do sapato na mesa;
- Identificação da sola do sapato para reprodução do seu modelo tridimensional;
- Criação do modelo tridimensional do sapato a partir dos pontos vindos da identificação da sola do sapato *RobotStudio*;
- Geração do programa usufruindo do pacote *Machining PowerPac*;
- Exportação do programa e realização do teste de acabamento da sola do sapato.

6.1 Configuração da Célula Robotizada para Experimentação

A célula robotizada utilizada para a realização de testes, ilustrada na Figura 6.1, é a célula existente no laboratório de robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto.
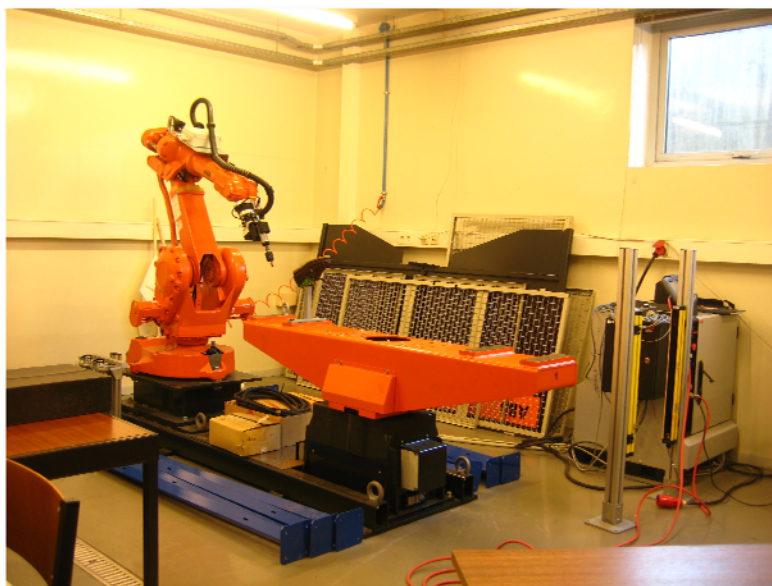
Figura 6.1 – Configuração real disponível no laboratório de robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto

Esta célula (Figura 6.1) possui uma configuração semelhante à solução número 2 proposta no tópico 3.5.1 do capítulo três, no entanto, esta encontra-se limitada à incorporação de dois sapatos de cada vez na sua mesa.

O princípio de funcionamento desta célula conta com que o robô, equipado com uma ferramenta, se desloque sobre o sapato que irá ser acabado. Posteriormente, a mesa rodará permitindo a entrada de um novo sapato, previamente colocado durante a operação de acabamento.

Os recursos existentes para a célula robotizada apresentada na Figura 6.1 são:

- Robô Industrial antropomórfico de seis eixos, *ABB IRB 2400*, com capacidade de carga no seu punho de dezasseis quilogramas e com um metro e meio de alcance máximo;
- Mesa rotativa *IRBp*, considerada como o sétimo eixo do robô;
- Dispositivo de controlo de força *ATI DELTA IP60*[26]*;*
- Ferramenta de corte e respectivo accionamento;

[26] Trata-se de um dispositivo capaz de medir seis componentes, três de força e três de binário, que consiste num transdutor equipado com um sistema de aquisição de dados inteligente.

- Controlador *IRB 5* com o *software RobotWare 5.10* e *RobotWare Machining 1.0;*
- Pilares de detecção de passagem da zona de trabalho.

As ferramentas de corte e respectivo accionamento não foram de facto objecto de estudo neste trabalho, no entanto, a finalidade de se testar a solução proposta, que visa a demonstração da sua aplicabilidade, levou à utilização dos recursos disponíveis, que consistiram:

- Ferramenta – ferramenta vinda de um robô industrial *SCARA*, onde foi construída uma interface de modo a ser possível adaptá-la ao robô *ABB* (Figura 6.2);
- Accionamento – motor pneumático alimentado à pressão disponível na rede (teoricamente de seis *Bar*), com uma velocidade de vinte e duas mil rotações por minuto (Figura 6.2);
- Ferramenta de corte:
    - Lixa cilíndrica de vinte milímetros de diâmetro (Figura 6.3 - A);
    - Esmoril cilíndrico de vinte e cinco milímetros de diâmetro (Figura 6.3 - B).
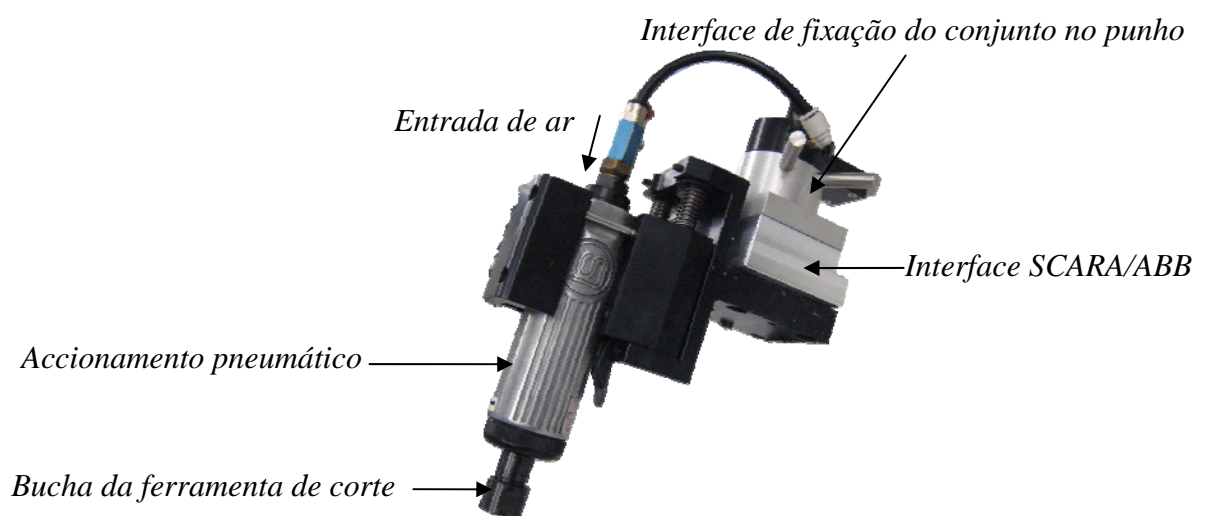
Figura 6.2 - Conjunto suporte e ferramenta do robô *ABB*
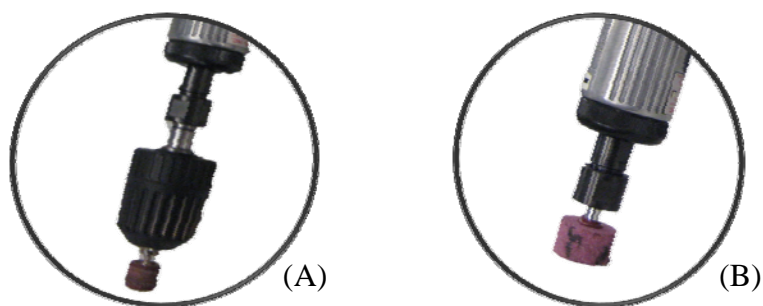
Figura 6.3 - Ferramentas de corte utilizadas (A – Lixa cilíndrica; B – Esmoril cilíndrico)

6.2 Fixação do Sapato

O sistema de fixação correcta do sapato não foi alvo de estudo neste trabalho, no entanto, é importante salientar a sua extrema importância.

Durante os vários processos de fabrico de um sapato, este transporta dentro de si uma forma, normalmente em material polimérico, conforme ilustrado na Figura 6.4. No que respeita ao acabamento de solas duplas, as suas principais funções são: conferir a forma correcta do sapato e oferecer a rigidez suficiente que simplifique o seu manuseamento durante a operação.

Figura 6.4 – Exemplo de forma do sapato

Os sapatos que foram utilizados para a implementação de testes foram os seguintes:

- Sapato clássico de homem, número quarenta e um de sola em couro (Figura 6.5 - A);
- Sapato de vela, número trinta e nove de sola dupla de borracha (Figura 6.5 - B).

Figura 6.5 – Sapatos utilizados para testes (A – Sapato clássico de homem; B – Sapato de vela)

A fixação dos sapatos à mesa de trabalho foi conseguida através do auxílio de um suporte, ilustrado na Figura 6.6.
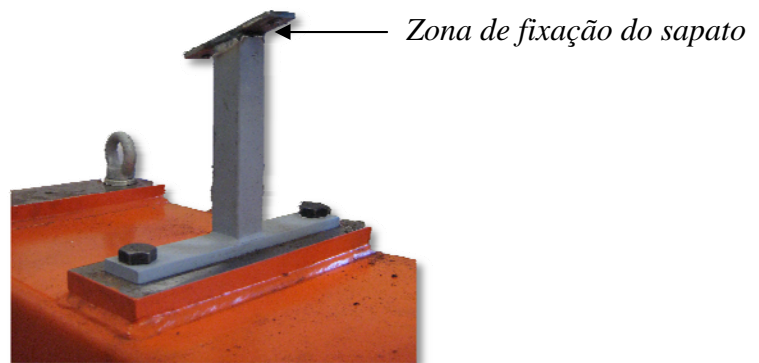
Figura 6.6 – Suporte para fixação do sapato

A existência do conjunto sapato de vela e respectiva forma, permitiu que fosse fixada a forma do sapato ao suporte, ilustrada na Figura 6.7.

Figura 6.7 – Sapato de vela fixado no suporte improvisado

A fixação do sapato clássico de homem foi menos elaborada, uma vez que não se dispunha da sua respectiva forma. Deste modo foi necessário realizar dois furos na sola para fixar o sapato, conforme ilustrado na Figura 6.8.
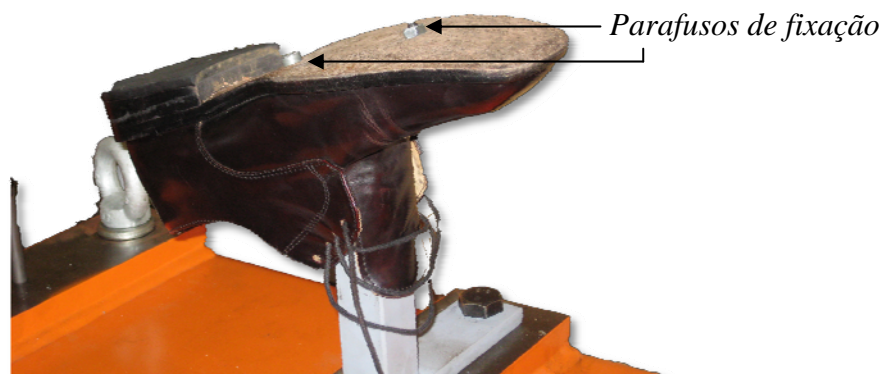
Figura 6.8 – Sapato clássico de homem fixado no suporte improvisado

Apesar da pouca flexibilidade das fixações existentes em ambos os testes, estas mostraram-se bastante satisfatórias nos propósitos de manter os sapatos imóveis durante os testes efectuados.

6.3 Identificação da Sola dos Sapatos

A identificação da localização da sola dos sapatos consiste num processo capaz de adquirir informação sobre este, onde posteriormente exista a possibilidade de a converter numa modelação tridimensional, sobre a qual incidirá a trajectória a programar. Este processo é comummente conhecido por engenharia inversa.

No mercado existem várias soluções de engenharia inversa, que vão desde digitalizadores tridimensionais a laser, até máquinas de medição de coordenadas por toque.

No âmbito do presente trabalho, não existiu qualquer intuito de realizar um estudo aprofundado sobre estes processos. No entanto, foi necessário recorrer aos meios disponíveis de modo a identificar a posição das superfícies a trabalhar, para criação de um modelo de *CAD* tridimensional da sola do sapato.

A solução encontrada consistiu no uso do próprio robô como máquina de medição de coordenadas. O procedimento de obtenção destas coordenadas foi o seguinte:

1. Criação de um programa na própria consola, utilizando a aplicação *RobotWare Machining*;
2. Movimentação do robô sobre os pontos de interesse da sola, através da activação prévia da tecnologia de controlo de força (Figura 6.9). No caso do

sapato de vela, foram medidos pontos no limite entre a sola inferior e superior, por apresentar um perfil constante e mais semelhante ao da forma final. No caso do sapato clássico de homem, foram medidos pontos entre o limiar do sapato e do corte do sapato, por este se apresentar já finalizado.

3. Exportação do programa em *RAPID* para o computador.

Figura 6.9 – Processo de identificação da sola do sapato

Após efectuada a exportação do programa para o computador foi necessário carregá-lo para o *RobotStudio*. Este procedimento revelou-se impraticável quanto à sua inserção directa, uma vez que o código do programa se encontrava apoiado em instruções de movimento em controlo de força.

A forma encontrada de contornar este problema, passou pelo desenvolvimento de uma aplicação dedicada (Figura 6.10) que permite transformar adequadamente o código do programa, num outro, passível de ser lido pelo *RobotStudio*.
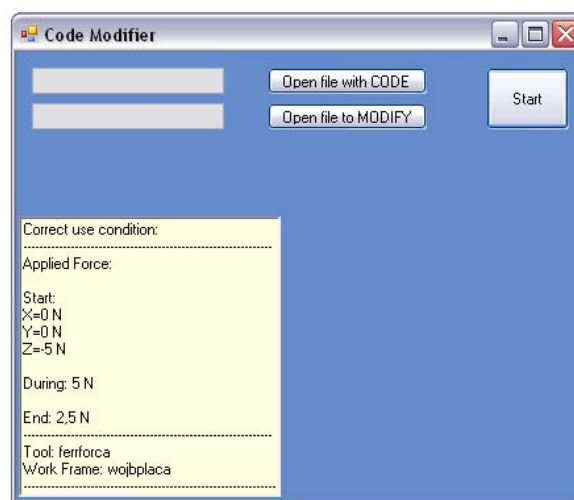
Figura 6.10 – Estrutura de funcionamento do programa de manipulação de código

O funcionamento desta aplicação envolve a realização dos seguintes procedimentos:

1. Selecção do ficheiro (módulo do programa) onde se encontra inserida toda a informação referente às características da sola (tamanho, forma, orientação) (botão *Open file with CODE*);
2. Selecção de um ficheiro, sem qualquer conteúdo, onde possa ser escrito o novo código (com as devidas modificações) (botão *Open file to modify*);
3. Iniciação da escrita do novo código (botão *Start*).

É importante referir que o uso correcto desta aplicação pressupôs que na realização do programa inicial (usando o próprio robô) se considerassem activos certos parâmetros, conforme ilustrado na Figura 6.10. Estas considerações foram realizadas de modo a permitir um desenvolvimento rápido desta aplicação, uma vez que esta apenas foi concebida para uso expedito durante a experimentação prática, não tendo existido qualquer objectivo no trabalho de desenvolver uma ferramenta deste tipo.

6.4 Criação do Modelo Tridimensional da Sola do Sapato

A modelação tridimensional do sapato necessitou da criação de um novo projecto no *RobotStudio*, onde se denominou a ferramenta e o referencial de trabalho de acordo com a situação real. Isto permitiu que ao inserir o módulo manipulado, se obtivesse uma concordância imediata entre a situação real e virtual.

O procedimento utilizado, que permite incorporar o módulo referente à informação espacial da sola do sapato, foi a seguinte:

1. Criação de um programa usando controlo de trajectória;
2. Sincronização desse programa com o controlador virtual;
3. Substituição do módulo onde se encontra disponível toda a informação das coordenadas pelo módulo manipulado, obtendo assim de forma imediata a nuvem de pontos do sapato (Figura 6.11);
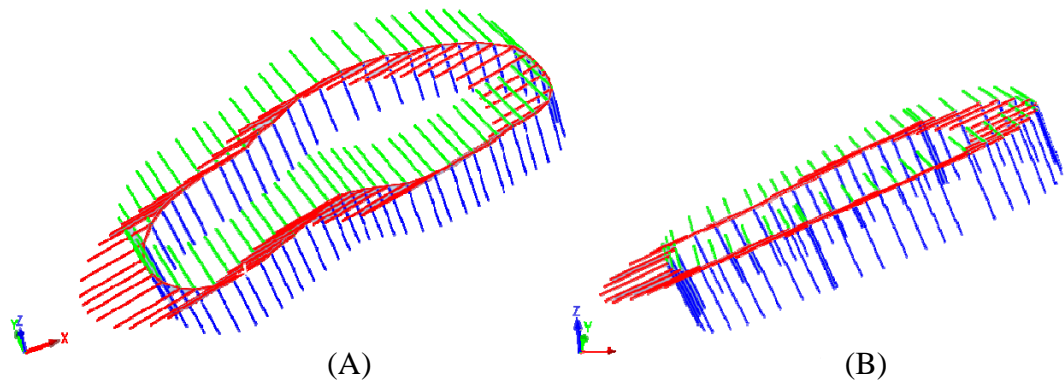
Figura 6.11 – Nuvem de pontos dos sapatos (A – Sapato Clássico de homem; B – Sapato de vela)

4. Reiniciação do controlador virtual na opção *I-START* (de modo a apagar todas as informações presentes no controlador virtual);
5. Eliminação de todas as trajectórias e pontos de desinteresse;
6. Criação de uma *spline*[27] utilizando as coordenadas do sapato (Figura 6.12);
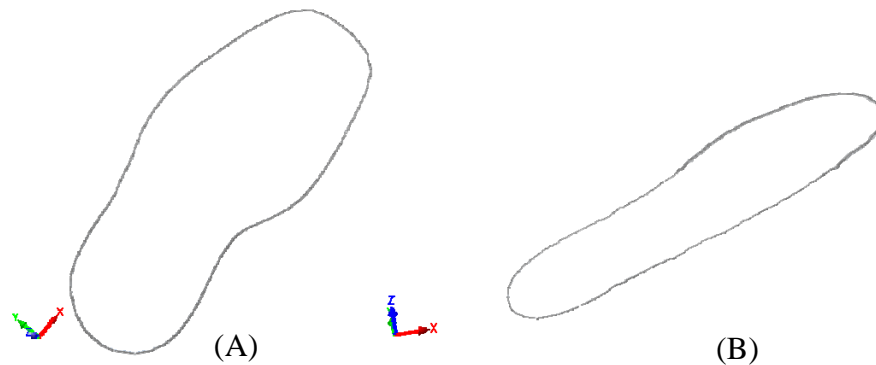
Figura 6.12 – *Splines* dos sapatos (A – Sapato clássico de homem; B – Sapato de vela)

7. Replicação de uma coordenada de modo a obter uma direcção para extrusão (*offset* referente ao seu eixo dos *ZZ*) (Figura 6.13);
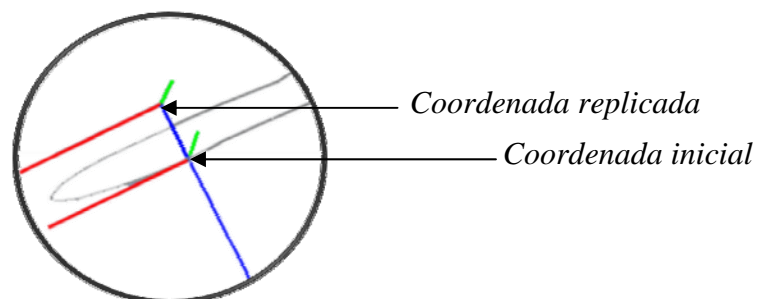
Figura 6.13 – Pormenor da replicação de uma coordenada

[27] Uma *spline* é definida como uma curva definida matematicamente por dois ou mais pontos de controlo.

8. Extrusão dessa *spline* na direcção constituída pelos dois pontos (original e replicado) (Figura 6.14).
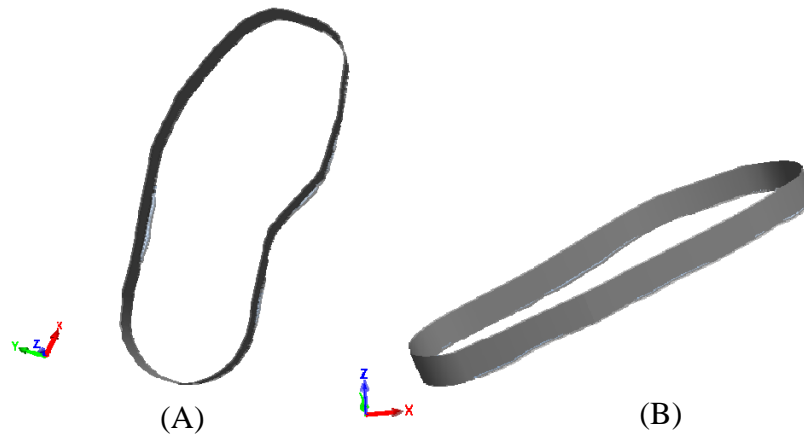
Figura 6.14 – Extrusão dos sapatos (A – Sapato clássico de homem; B – Sapato de vela)

6.5 Geração da Trajectória

A geração da trajectória baseou-se na metodologia proposta no capítulo cinco, onde, para as fases seguidamente apresentadas, constaram as seguintes considerações:

1. ***Definição dos parâmetros do processo de maquinagem***
    - Escolha do tipo do processo de maquinagem *FC Pressure;*
    - Direcção de aplicação da força relativamente ao eixo dos *YY* da ferramenta;
    - Força de encosto inicial de *10 N*;
    - Força durante o processo de *10 N*;
    - Força de afastamento de *5 N*;
    - Velocidade de *500 mm/s* de encosto e afastamento;
    - Velocidade de *100 mm/s* durante o processo.

A escolha do processo de maquinagem *FC Pressure*[28], em detrimento da função *FC SpeedChange*, baseou-se na impossibilidade de garantir que a localização da sola do sapato se mantenha sempre inalterável. Assim, a utilização do processo *FC Pressure*, garantiu que o robô se adaptasse à sola dos sapatos, mostrando-se a solução mais adequada.

Os parâmetros acima definidos, tanto para a força como para a velocidade, foram estipulados com valores baixos, devido à inexistência de equipamento preparado para a utilização no processo de corte. Garantiu-se, desta forma, a existência de alguma segurança quer do equipamento, quer do grupo humano de trabalho.

2. ***Escolha da trajectória***

A escolha da trajectória para realização da operação constou na selecção da superfície inferior da modelação tridimensional por esta significar o limite a não ultrapassar da sola.

3. ***Definição da orientação dos pontos da trajectória***

A definição da trajectória foi definida como sendo um conjunto de pontos e orientações perpendicularmente à superfície da modelação, de modo a garantir a uniformidade do acabamento.

Contrariamente à consideração feita na simulação do acabamento (capítulo cinco), em que não existiram preocupações relativamente ao sentido da trajectória, no caso prático, foi necessário considerar a mesma. A atribuição deste sentido foi baseada sobre o tipo de ferramenta de corte e respectivo accionamento. Desta forma, foi considerado que o sentido da trajectória se teria de realizar no sentido indirecto, devido à existência de ferramentas abrasivas accionadas no sentido indirecto, sendo que, neste sentido a sola iria ser cortada e comprimida, o que se traduz num melhor acabamento final.

Antes de se proceder ao sincronismo da solução com o controlador virtual do robô, foi efectuado um deslocamento de dois milímetros segundo o eixo dos *ZZ* da trajectória do sapato de vela no sentido do corte do mesmo. Foi necessário realizar este deslocamento de modo a colocar a trajectória no limiar da sola e do corte do sapato, devido à abordagem que se teve na medição das coordenadas da sola.

---

[28] Ver tópico 5.2.1 do capítulo cinco para relembrar este conceito de operação.

6.6 Realização de Testes de Acabamento em Solas de Sapatos

Após efectuado o sincronismo das soluções propostas com o controlador virtual, fez-se a devida exportação destas, em forma de programa *RAPID* para o controlador real.

O processo de corte dos sapatos envolveu em primeiro lugar, uma pré-execução do programa sem que a ferramenta estivesse activa, utilizando uma velocidade de trajectória de 10 % sobre a programada[29]. O objectivo desta pré-execução consistiu na visualização do comportamento do robô face à trajectória programada.

Verificadas estas condições, executaram-se os processos de corte de ambos os sapatos utilizando ambas as ferramentas (lixa e esmoril). Estes processos foram executados a uma velocidade de 10 % da velocidade implementada (por motivos de segurança).

Por fim, testou-se a solução a uma velocidade maior (40 % da velocidade implementada), de modo a obter-se uma aproximação dos tempos reais do processo de corte.

6.6.1 Acabamento da Sola do Sapato Clássico de Homem

Os resultados obtidos no acabamento do sapato clássico de homem revelaram-se fracos com a aplicação do esmoril. Esta ferramenta não foi capaz de realizar o devido corte de material em boas condições, resultando num acabamento de aspecto não homogéneo, conforme ilustrado na Figura 6.15.

Figura 6.15 – Pormenor de acabamento do sapato com a utilização do esmoril

[29] Relembrando: a velocidade programada foi de cem milímetros por segundo.

Contrariamente à situação verificada pela utilização do esmoril, a lixa desempenhou satisfatoriamente o acabamento, obtendo-se uma delineação da sola do sapato sem oscilações e um acabamento homogéneo. No seguinte conjunto de figuras (Figura 6.16, Figura 6.17, Figura 6.18 e Figura 6.19), encontra-se ilustrado o referido acabamento.
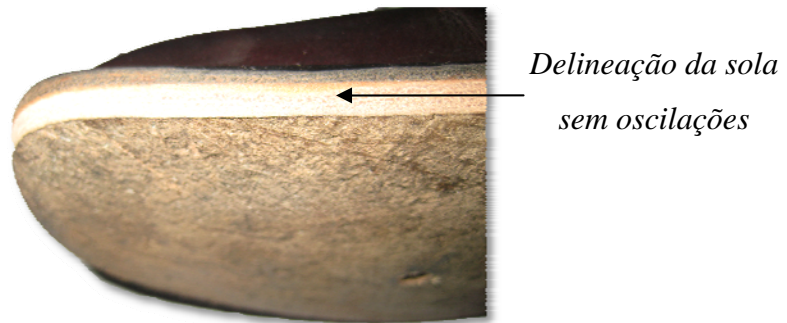
Figura 6.16 – Pormenor de acabamento do sapato com a utilização da lixa
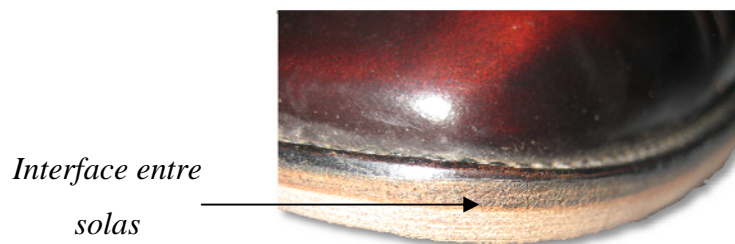
Figura 6.17 – Acabamento da zona da biqueira do sapato clássico de homem

Figura 6.18 – Acabamento visto de baixo do sapato clássico de homem

Figura 6.19 - Aspecto geral do sapato clássico de homem após acabamento

Na Figura 6.19, torna-se visível que a zona do tacão não se encontra acabada. Esta situação deve-se ao excessivo tamanho da bucha de aperto da lixa, que, ao passar naquela zona, fazia com que apenas a bucha se encostasse, não permitindo que a lixa trabalhasse correctamente.

6.6.2 Acabamento da Sola do Sapato de Vela

O acabamento da sola do sapato de vela revelou-se num processo inconclusivo. As ferramentas de corte utilizadas revelaram-se inapropriadas para o caso em estudo, chegando a lixa a tornar-se inutilizável devido ao excessivo calor envolvido no processo de corte (Figura 6.20).

Figura 6.20 – Lixa cilíndrica danificada durante o processo de corte

A sola de borracha necessita de uma ferramenta com um accionamento mais potente e de uma ferramenta mais eficaz (como por exemplo uma fresa).

7 Conclusões e Trabalhos Futuros

7.1 Problemas Encontrados

Durante esta investigação foram surgindo complicações e problemas que devem ficar registados para permitir uma melhor análise de todo o processo e que justificam algumas das opções tomadas e soluções abandonadas.

Primeiramente, e do ponto de vista teórico, é importante referir que procura de soluções robotizadas para efectuar acabamentos em sapatos revelou-se numa tarefa árdua, dada a escassez de informação existente (essencialmente em suporte digital). Relativamente a células robotizadas usufruindo de controlo de força, não foram encontradas quaisquer referências na área de estudo deste trabalho, talvez por se tratar de uma tecnologia recente e em expansão.

Desta forma, após efectuado um levantamento acerca dos processos necessários para a compreensão do modo de simular uma célula robotizada, descritos ao longo deste documento, foi realizada a simulação do processo de corte de solas duplas no *software RobotStudio*, utilizando uma configuração da célula robotizada idêntica à disponível no laboratório de robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto. A simulação foi conseguida recorrendo ao uso do *software Maching PowerPac*, apoiada no uso do controlo de força, revelando-se uma boa ferramenta de apoio à concepção de um programa na linguagem do robô: *RAPID*. No entanto, apenas é possível efectuar uma simulação cinemática do processo (apesar de baseado no controlo de força), o que impede a total validação da simulação do processo de acabamento.

O uso do controlador virtual permitiu a criação e verificação de um programa em *RAPID* de forma expedita. Ainda assim, apesar de ter sido uma ferramenta importante, apresentou alguma instabilidade. Verificou-se, também, que no decorrer da simulação de

movimentos de interpolação de juntas, por vezes o controlador virtual tem dificuldades em cumprir o processo.

Como requisito do trabalho, no sentido de se permitir a execução do processo de acabamento de sapatos por pessoal não qualificado, foi concebida e implementada uma aplicação gráfica para a consola do robô, cuja aplicação na consola virtual do *RobotStudio* se revelou como perfeitamente funcional (ignorando os erros do controlador virtual, apresentados anteriormente), operando sem qualquer problema. Contrariamente ao sucesso verificado na aplicação virtual, e apesar da implementação e do arranque terem decorrido sem problemas, a aplicação real revelou dificuldades ao aceder a alguns menus que necessitavam de leitura directa da base de dados, devido ao conflito entre versões de *software*. Verificaram-se, ainda, algumas dificuldades na leitura do eixo externo do robô (mesa rotativa), devidas a uma falha na função capaz de efectuar a leitura do eixo (erro confirmado no fórum da *ABB*[30]). Este problema foi contornado recorrendo à activação prévia da mesa rotativa, imediatamente antes de se proceder à sua rotação. Desta forma, foi possível ler o seu eixo, como se se tratasse de um eixo de um robô.

Detectados e contornados os problemas, procedeu-se à realização de testes laboratoriais em dois tipos de sapatos de sola dupla (vela e clássico de homem), que apesar de não terem ocorrido em condições ideais, devido à inexistência de estudos do processo de corte e de um sistema de fixação adequado (os quais não constavam no estudo do corrente trabalho), levaram à obtenção de informações importantes no estudo desta solução. No caso processamento do sapato clássico de homem, foi obtido um acabamento uniforme, sem oscilações na trajectória.

Finalmente, com o objectivo de complementar este trabalho, foi testada a utilização do *RobotWare Machining* presente na consola real do robô, com o intuito de verificar se seria possível usufruir desta ferramenta para realizar o referido acabamento. De facto, foi possível ensinar a trajectória da sola ao robô, a qual inclusivamente foi praticável pelo facto de o robô optimizar essa trajectória numa busca automática em controlo de força sobre a sola. No entanto, quando se implementou a solução obtida, constatou-se que a orientação dos referenciais de cada ponto da trajectória se manteve sempre inalterada, não cumprindo assim a direcção perpendicular à superfície, como tinha sido perspectivado. A solução encontrada para resolver este problema seria a de alterar todas as orientações dos pontos ensinados,

---

[30] Erro reportado em *http://www.robotstudio.com/forum/ forum_posts.asp?TID=1993, visto em 20/06/09*

instrução que seria temporalmente dispendiosa. Neste campo, a utilização da consola real para realizar operações de maquinagem, não se revelou como uma boa ferramenta, pelas razões mencionadas.

7.2 Conclusões

O objectivo deste trabalho foi o de conceber e simular uma célula robotizada que permitisse realizar o acabamento automatizado de sapatos de solas duplas, baseado na utilização de um robô industrial equipado com controlo de força.

Neste sentido, após o desenrolar de todo o processo e apesar dos problemas encontrados ao nível da procura de soluções robotizadas, foi possível efectuar a simulação de uma célula em contexto laboratorial que responde aos seguintes requisitos propostos inicialmente:

- Capacidade de acomodar diferentes modelos de sapatos;
- Capacidade de adaptação aos diferentes tamanhos de sapatos;
- Uniformidade do acabamento;
- Uma interface de comando intuitiva que permita uma fácil operação da célula por parte de um utilizador não especializado.

Com o realizar desta simulação, foi possível perceber que a aplicação de uma solução robotizada pode melhorar a performance do sector de acabamento de sapatos na indústria nacional, pelos motivos listados:

- *Redução no tempo de realização do acabamento* - Comparativamente aos, aproximadamente, quinze segundos[31] dispendidos por um operador experiente na operação de acabamento da sola de um sapato, o estudo permitiu verificar que um robô é capaz de realizar a mesma tarefa em sensivelmente oito segundos, ou seja, existe redução em cerca de metade do tempo;

---

[31] Valor médio verificado na visita à empresa Armipex

- *Aplicação para a consola intuitiva* - A aplicação desenvolvida permite que um operador possa iniciar de um modo intuitivo o processo de acabamento de solas, seleccionando as características do sapato presente na célula (tipo, número, e orientação do pé). Isto é possível devido à existência de uma base de dados, previamente programada e introduzida por um programador, que permite a incorporação de toda a informação relativa ao calçado existente numa empresa. A existência de uma base de dados na aplicação desenvolvida permite que esta possa ser implementada para operação de qualquer configuração de célula robotizada para operações de acabamento de calçado;
- *Operadores não especializados* - A qualificação de operadores deixa de ser necessária, resumindo-se à leitura e compreensão do manual do utilizador.

Desta forma, tornou-se possível compreender que a introdução duma solução robotizada deste tipo em empresas de calçado poderá auxiliar no combate à quebra do processo produtivo, facto que se pretendia aferir com a realização desta investigação.

Com efeito, este tipo de soluções, quando aplicadas com sucesso, podem, não só auxiliar no processo de acabamento de solas de sapatos, como também reduzir o tempo dispendido no mesmo e contrariamente ao verificado até então, a aplicação de um *software* de simples utilização leva a que não exista necessidade de operação por parte de trabalhadores especializados.

Conclui-se assim, que este tipo de soluções robotizadas, disponíveis durante vinte e quatro horas e que podem trabalhar com químicos perigosos (Abreu, 2008), possibilita a uma empresa poupar ao nível dos custos e recursos (Kim, 2004), permitindo que esta invista numa maior panóplia de sapatos e, consequentemente, possa competir no mercado nacional face à oferta de empresas estrangeiras.

7.3 Trabalhos Futuros

A fim de se concretizar devidamente o estudo presente neste trabalho, deverão ser realizados estudos acerca das ferramentas a utilizar no processo de corte de sapatos de sola dupla e respectivo accionamento. Igualmente importante será a realização de um estudo aprofundado, com consequente concepção, de um sistema de fixação automático de sapatos.

Seria também imperativo levar a cabo um estudo que permita a obtenção, de uma forma automática, do tipo, do tamanho e da orientação de sapatos. Como complemento deste estudo deverá ser englobada a possibilidade de extrapolar um dado tipo de sapato para todos os outros números. Desta forma tornar-se-á possível realizar apenas o levantamento de um tipo de sapato e realizar a adaptação para os restantes números pertencente ao mesmo tipo, funcionalidade que, no presente trabalho, foi realizada pela programação individual de todos os sapatos.

Por fim, deverá ser levada a cabo uma melhoria das capacidades da consola a fim de integrar melhor as limitações sentidas na implementação da aplicação desenvolvida.

8 Bibliografia

ABB. *ABB Robotics.* [Online] www.abb.com/robotics.

—. *ABB Robotics.* [Online] www.abb.com/review.

—. 2007. Force Control for Machining. *Application Manual.* Revision B, 2007.

—. 2008. Machining PowerPac. *Operating Manual.* 2008.

—. 2007. RAPID. *Reference Manual.* 2007.

—. 2006. Robot Application Builder. *Application Manual.* Revision B, 2006.

—. 2002. *RobotStudio.* Maio de 2002.

Abreu, Paulo. 2001. *Aplicaçações industriais de robôs.* Faculdade de Engenharia da Universidade do Porto. Porto : s.n., 2001. Textos de Apoio.

—. 2008. *Apontamentos de robótica.* Faculdade de Engenharia da Universidade do Porto. Porto : s.n., 2008. Textos de Apoio.

Brorsson, Angela, Sjöberg, Ralph e Liberg, Anna. 2006. Do it yourself robotics, embedded software allows users to program their own robot application. *ABB Review.* 2006, Vol. 2.

Camarotto, João Alberto. 1998. *Estudo das relações entre o projeto de edifícios industriais e a gestão da produção.* Faculdade de Arquitectura e Urbanismo de São Paulo. São Paulo : s.n., 1998. Tese de Doutoramento.

CEGEA. 2007. *Plano Estratégico 2007-2013 para a Industria do Calçado.* s.l. : APPICAPS, 2007.

Chan, S.F. e Kwan, Reggie. 2003. Post-processing methodologies for off-line robot programming within computer integrated manufacture. *Journal of Materials Processing Technology 139* . 2003, pp. 8–14.

Desma. Desma. [Online] www.desma.de.

GMBH, Klockner Desma Schuhmaschinen. 2007. Desma Shoe Machinery. *Desma Today.* 2007, 04.

Groover, Mikell P. 1996. *Industrial robotics : technology, programming, and applications.* New York : McGraw - Hill Book Company, 1996.

Hollingum, Jack. 1996. Quick-Change automation for shoe manufacture. *Assembly Automation.* 1996, Vol. 16, pp. 34-39.

Hunt, V. Daniel. 1983. *Industrial Robotics Handbook.* New York : Industrial Press INC., 1983.

Kim, J. Y. 2004. CAD-Based Automated Robot Programming Adhesive Spray Systems for Shoe Outsoles and Uppers. *Journal of Robotics Systems.* 2004, Vol. 21, 11, pp. 625-634.

Kochan, Anna. 1996. Actis and the shoe industry. *Assembly Automation.* 1996, Vol. 16, 3, pp. 30-31.

Koren, Yoram. 1987. *Robotics for Engineers.* New York : McGraw-Hill book, 1987.

Limited, Emerald Group Publishing. 2006. European shoe industry looks to production and process innovation. 2006, Vol. 22, pp. 32-34.

McKerrow, Phillip John. 1991. *Introduction to Robotics.* Sidney : Addison-Wesley Publishing Company, 1991.

Nemec, Bojan, Lenart, Borut e Zlajpah, Leon. 2003. *Automation of Lasting Operation in Shoe Production Industry.* Jozef Stefan Institute. Ljubljana, Slovenia : IClT, 2003.

Robot System. 2009. *www.robotsystem.it.* [Online] 2009. www.robotsystem.it.

Sharp, John. 2008. *Microsoft Visual C# 2008 Step by Step.* Washington : Microsoft Press, 2008.

Spencer Jr, James E. 1996. Robotics technology and the advent of agile manufacturing systems in the footwear industry. *Assembly Automation.* 1996, Vol. 16, 3, pp. 5-15.

Vilaça, João L. e Fonseca, Jaime. 2007. *A new software application for footwear industry.* Industrial Electronics Department, University of Minho. Guimarães, Portugal : s.n., 2007.

ANEXOS

ANEXO A - Manual de Utilização da Aplicação Desenvolvida

Manual de Utilização

Footwear Finishing

Versão 1.0

Índice de Conteúdos

1 Introdução

O *software* denominado de *Footwear Finishing* é uma aplicação gráfica desenvolvida com o principal objectivo de proporcionar ao utilizador não experiente em robótica, uma forma intuitiva de executar o processo de corte de sapatos.

2 Requisitos

Ao longo deste ponto irão ser elencados os principais requisitos para funcionamento e aplicação do software em causa.

2.1 Consola Real

A instalação desta aplicação na consola real do robô requer a existência dos seguintes recursos:

- Robô *ABB* equipado com um controlador *ABB IRC 5*;
- Dispositivo de controlo de força;
- Versão do *RobotWare* igual ou superior à *5.10*;
- Qualquer versão do *RobotWare Machining*;

2.2 Consola Virtual

A instalação desta aplicação na consola virtual tem como exigências computacionais as mesmas que o *RobotStudio*, que deverá ser uma versão igual ou superior à 5.11.

As considerações a ter para que a aplicação seja instalada correctamente pressupõem que exista um projecto no *RobotStudio* suportado por um controlador virtual com uma versão igual ou superior à 5.10.

3 Instalação

A instalação da aplicação consiste na cópia de uma base de dados previamente definida e de dois ficheiros com extensão *“.dll”* denominados por *TpsViewFootWearFinishing* e *TpsViewFootWearFinishing.gtpu*.

3.1 Instalação da Base de Dados

A base de dados constituída por um conjunto de ficheiros e pastas, (ilustrado na Figura 1) deverá ser copiada para a raiz do dispositivo de armazenamento em questão (quer real quer virtual).
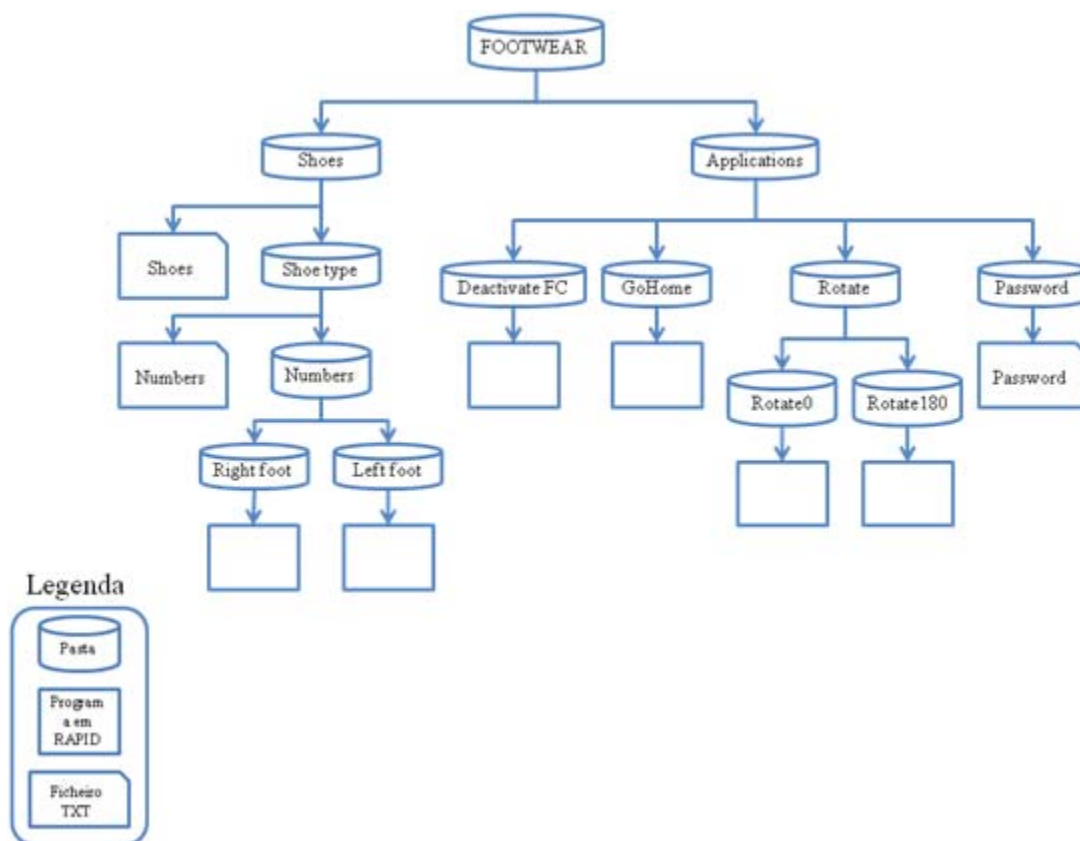
Figura 1 – Estrutura da base de dados

Previamente à instalação da base de dados apresentada, o utilizador deverá editar o seu conteúdo de modo a configurá-lo para a sua correcta aplicação. Esta edição, efectuada com recurso a qualquer editor de texto, deverá ser realizada nos ficheiros com extensão *“.txt”*: o ficheiro *Shoes* e o ficheiro *Numbers*. No primeiro, o utilizador deverá escrever todos os modelos de sapatos a incluir na base de dados. Posteriormente, o utilizador deverá criar tantas

pastas quantos os modelos inseridos no ficheiro *Shoes*. A denominação destas pastas deverá ser a mesma introduzida no ficheiro "*.txt*", no entanto, esta deverá ser realizada sem o uso de qualquer espaço ou caracteres especiais. Na Figura 2 poderá observar-se um exemplo deste procedimento.
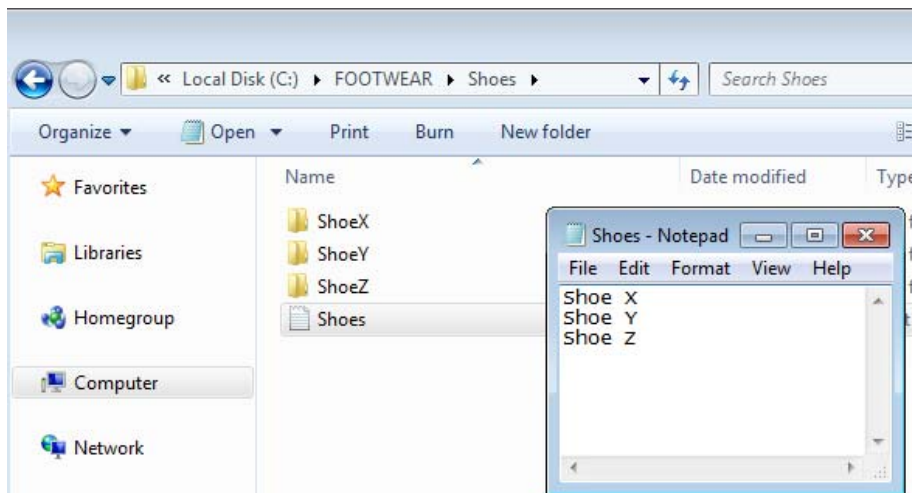
Figura 2 – Exemplo de edição e criação de pastas dos modelos dos sapatos

Olhando para o esquema da Figura 1, verifica-se que as pastas dos modelos dos sapatos deverão conter informações dos modelos. De modo análogo ao procedimento da criação de pastas de sapatos, é necessário proceder à criação de pastas para os números destes. No entanto, previamente a esta operação, deverá ser criado um ficheiro "*.txt*" na pasta de cada tipo de sapato existente, que abarcará a informação respeitante aos números disponíveis de cada modelo, contendo, ainda, o nome do modelo do sapato. Para finalizar, é feita a edição do ficheiro, que consistirá na introdução dos números de sapatos existente para cada respectivo modelo. Na Figura 3 encontra-se ilustrado um exemplo deste procedimento.
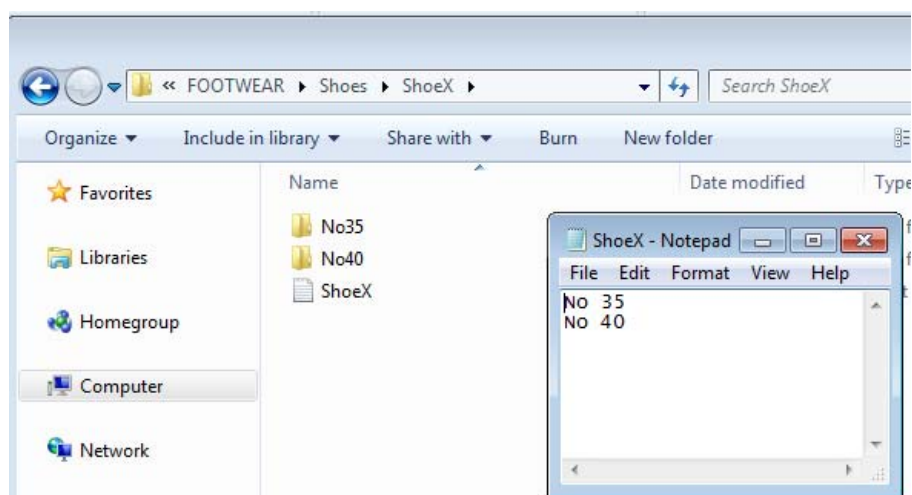
Figura 3 – Exemplo de edição e criação dos números dos sapatos para um dado modelo

O penúltimo procedimento da configuração da base de dados consiste na criação de duas pastas, *LeftFoot* e *RightFoot*, contida na pasta do respectivo número. Este procedimento pode ser observado na seguinte figura (Figura 4).
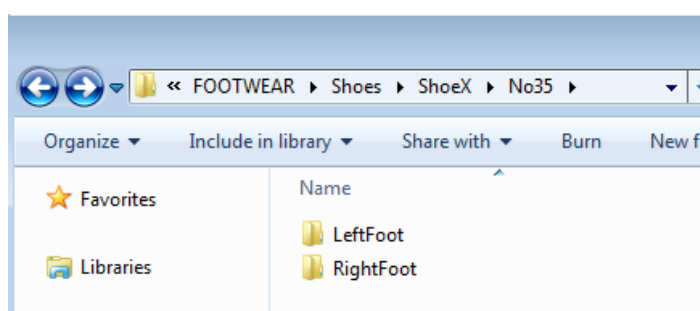
Figura 4 – Exemplo da criação das pastas *LeftFoot* e *RightFoot*

O último procedimento consiste na cópia dos programas de acabamento de sapatos em linguagem *RAPID* para as respectivas pastas, *LeftFoot* ou *RightFoot*, sempre com o nome *Shoe*. Como exemplo deste procedimento tem-se um sapato do modelo *Shoe X*, número trinta e cinco, do pé esquerdo. Desta forma, o respectivo programa *RAPID* deverá ser copiado para dentro do directório */FOOTWEAR/Shoes/ShoeX/No35/LeftFoot* (Figura 5).
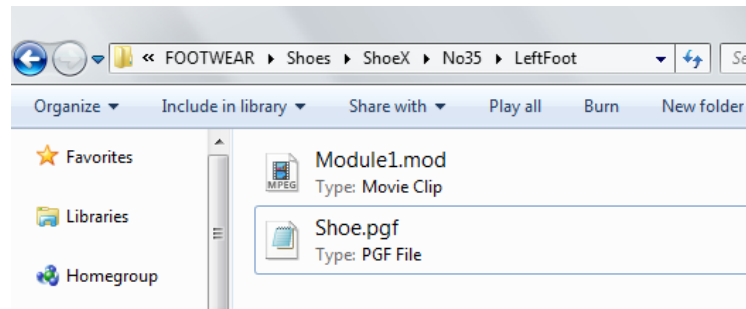
Figura 5 – Exemplo da presença dos ficheiros em linguagem *RAPID*

3.2 Instalação dos ficheiros “*.dll*” na Consola Virtual

A instalação da aplicação na consola virtual do robô requer que sejam copiados os dois ficheiros com extensão “*.dll*” apresentados anteriormente, para o directório *HOME* de um projecto com controlador virtual (Figura 6). Finda esta operação, é necessário reiniciar o controlador virtual.
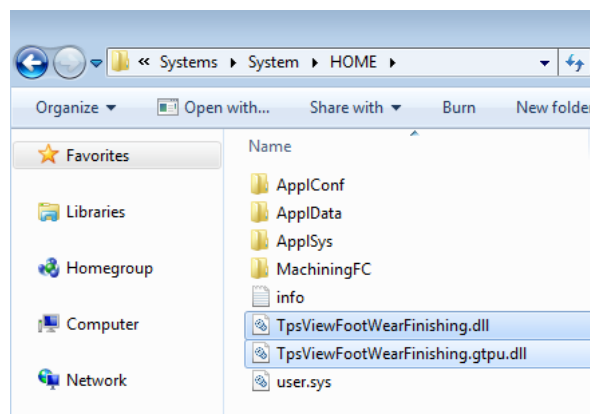
Figura 6 – Demonstração da localização dos ficheiros “*.dll*” após instalação

Nota: Copiar os ficheiros “*.dll*” destinados ao controlador virtual.

3.3 Instalação dos ficheiros “*.dll*” na Consola Real

A instalação da aplicação na consola real do robô deve ser realizada recorrendo ao auxílio da aplicação *Compliance Tool* (Figura 7), disponibilizada na pasta de instalação do *software Robot Application Builder 5.1X*. Para realizar esta operação, é necessário estabelecer uma ligação por rede com o controlador do robô. Posteriormente bastará seleccionar os dois

ficheiros com extensão “*.dll*”, indicando qual o endereço de *IP* com o qual comunicar. Seguidamente é necessário reiniciar o controlador real.
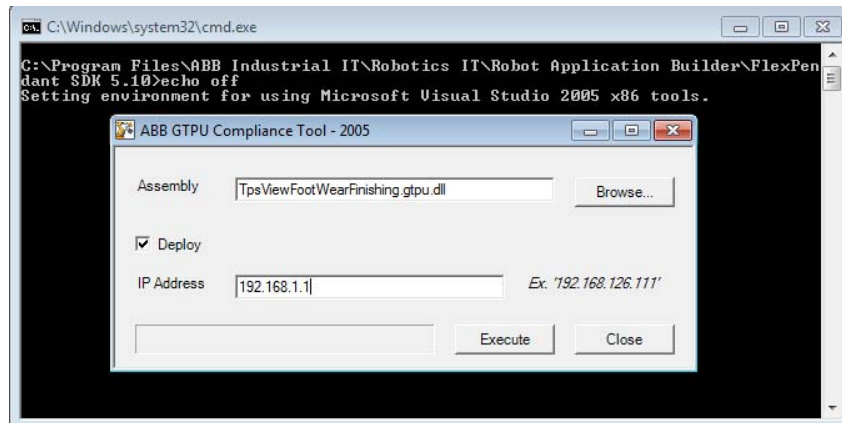
Figura 7 – Uso do *Compliance Tool* para instalação dos ficheiros “*.dll*”

Nota: Copiar os ficheiros “*.dll*” destinados ao controlador real.

4 Iniciar aplicação

4.1 Consola Real

O acesso à aplicação encontra-se disponível no menu principal da consola real do robô, pelo que bastará um simples clique para a iniciar (Figura 8).
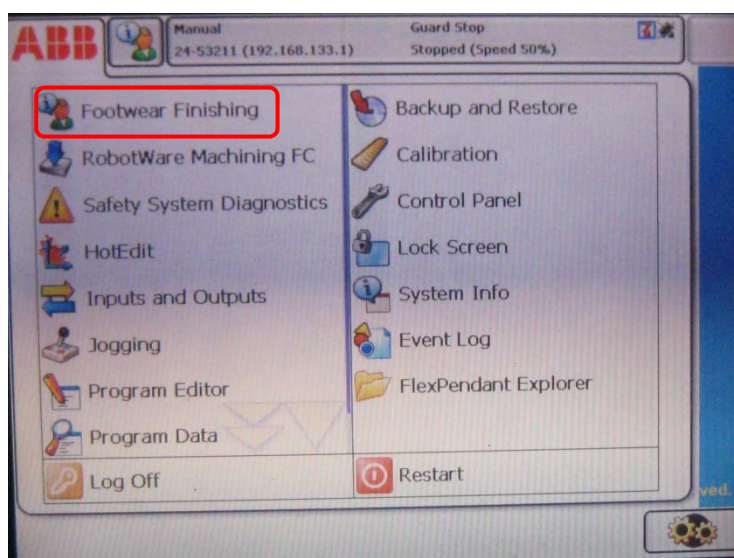
Figura 8 – Localização da aplicação *Footwear Finishing* na consola real

4.2 Consola Virtual

O acesso à aplicação encontra-se disponível no menu principal da consola virtual do robô. Esta intitula-se *Virtual FlexPendant* e apresenta-se na *tab Offline* disponível na barra de ferramentas (Figura 9). Para iniciar a aplicação basta proceder ao clique no local mencionado.
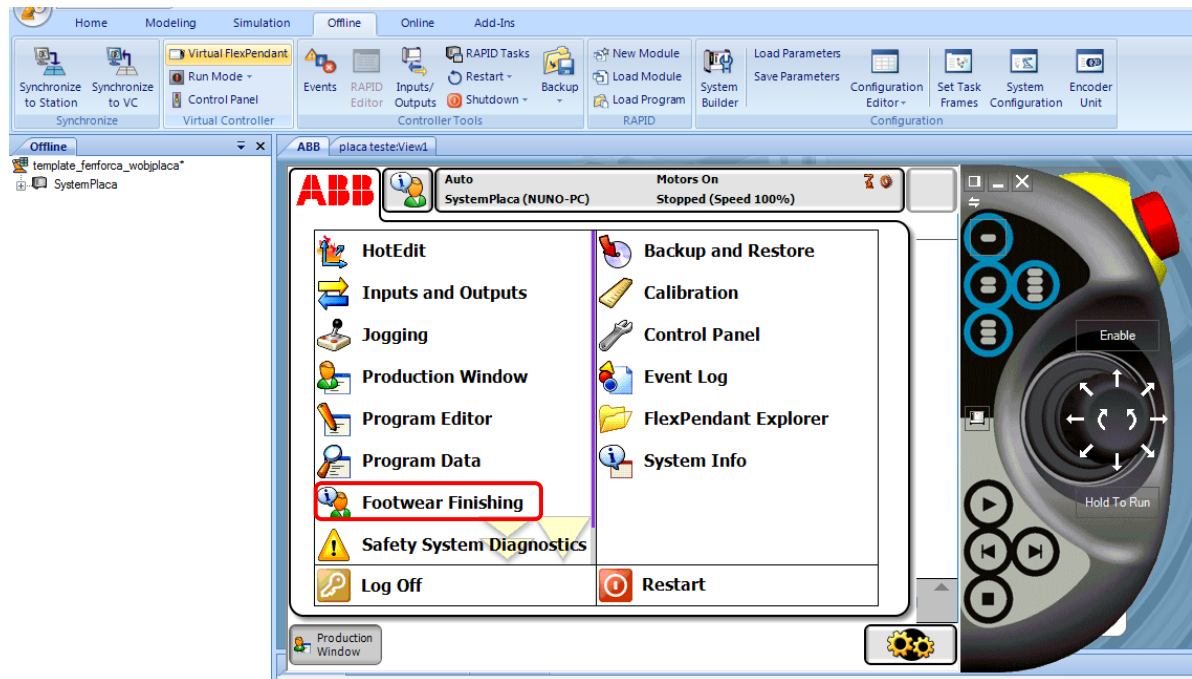
Figura 9 – Localização da aplicação *Footwear Finishing* na consola virtual

5 Funcionamento da aplicação

Após o clique de entrada no programa, será exibida uma imagem introdutória com informações sobre a aplicação (Figura 10).
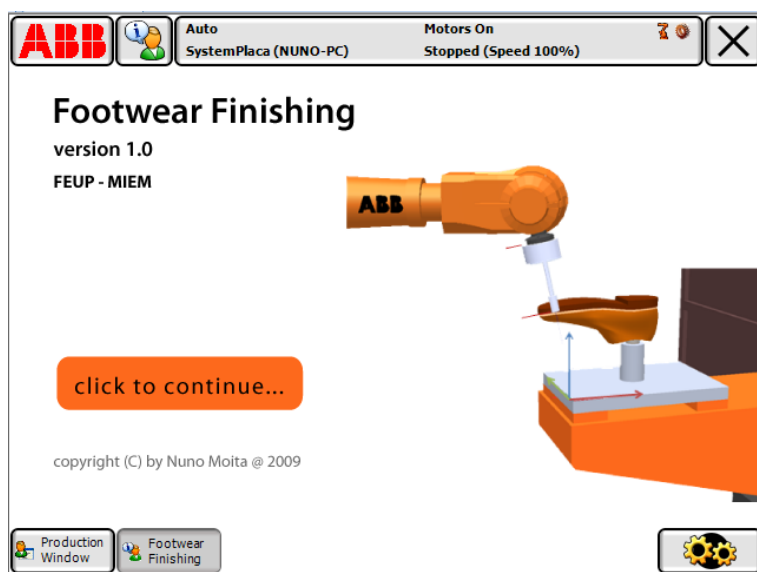
Figura 10 – Imagem introdutória à aplicação *Footwear Finishing*

5.1 Menu Inicial

O menu inicial, apresentado na Figura 11, é composto por três botões de acesso a submenus denominados:

- *Process Shoes;*
- *Load/Erase Shoes;*
- *DeBug.*
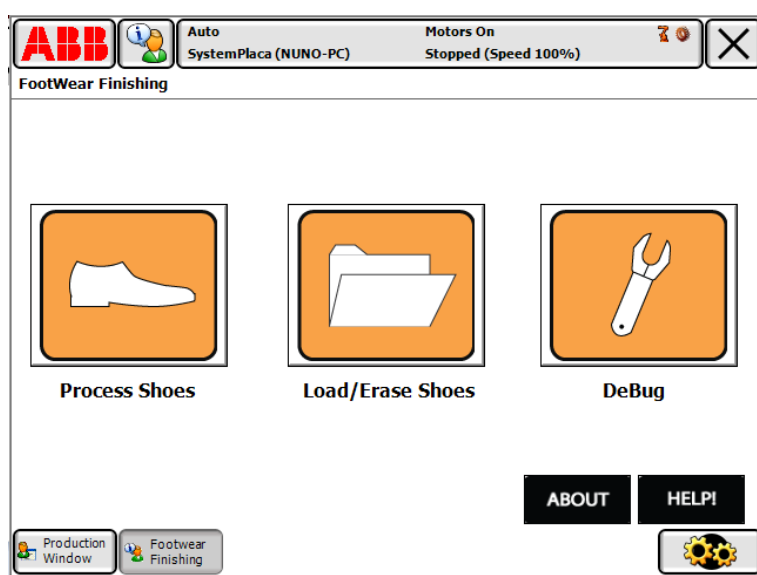
Figura 11 – Apresentação do menu inicial

Além destes botões, o menu principal contém outros dois: o *About* e o *Help*. O botão *About* disponibiliza informações de concepção, autoria e respectiva versão da aplicação (Figura 12).
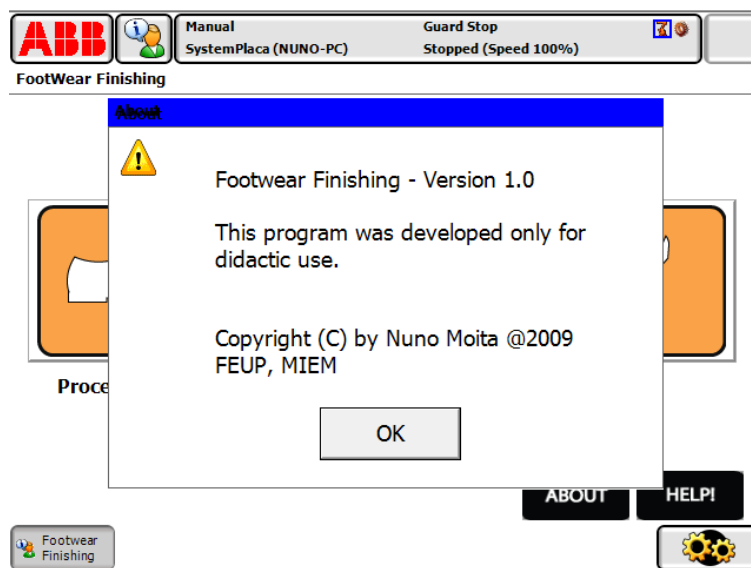
Figura B.1

Figura 12 – Informação disponível após clique do botão *About*

O botão *Help* pretende elucidar o utilizador, de uma forma breve e concisa, sobre o modo de funcionamento do programa. Este submenu é constituído por duas *tabs*: uma contendo informação sobre o modo de utilização das funcionalidades disponíveis no submenu *Process Shoes* (Figura 13), e outra sobre quando e como utilizar o submenu *DeBug* (Figura 14).
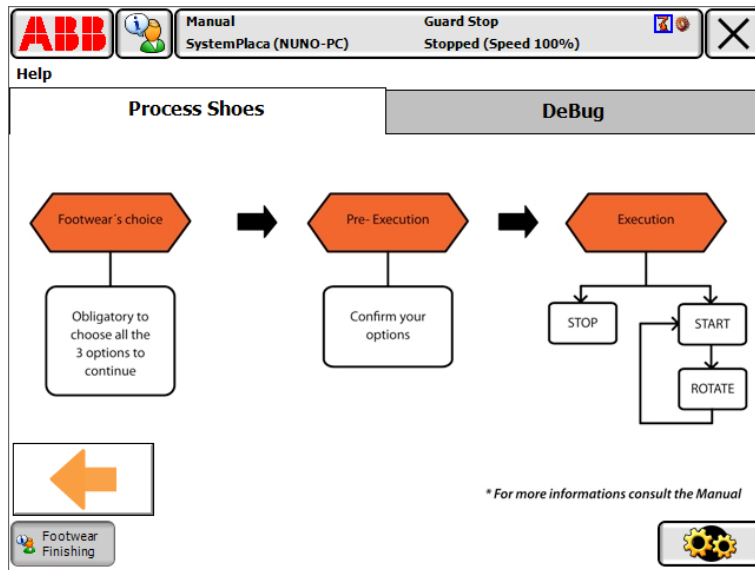
Figura 13 – Ajuda disponível na *tab Process Shoes*

Nota: Ver tópico 5.2 para informação mais detalhada.
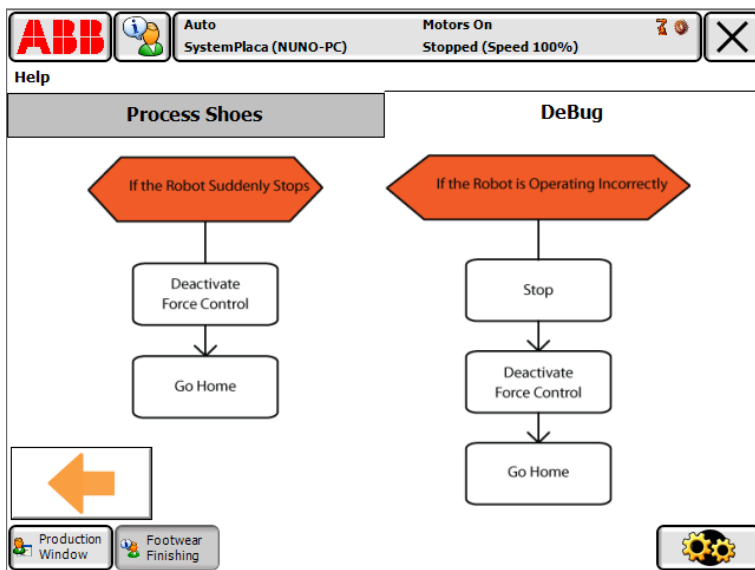
Figura 14 - Ajuda disponível na *tab DeBug*

Nota: Ver tópico 5.4 para mais informação.

5.2 Submenu *Process Shoes*

Este submenu (Figura 15) é constituído pelas seguintes três *tabs*:

- *Footwear's Choice;*
- *Pre-Execution;*
- *Execution.*
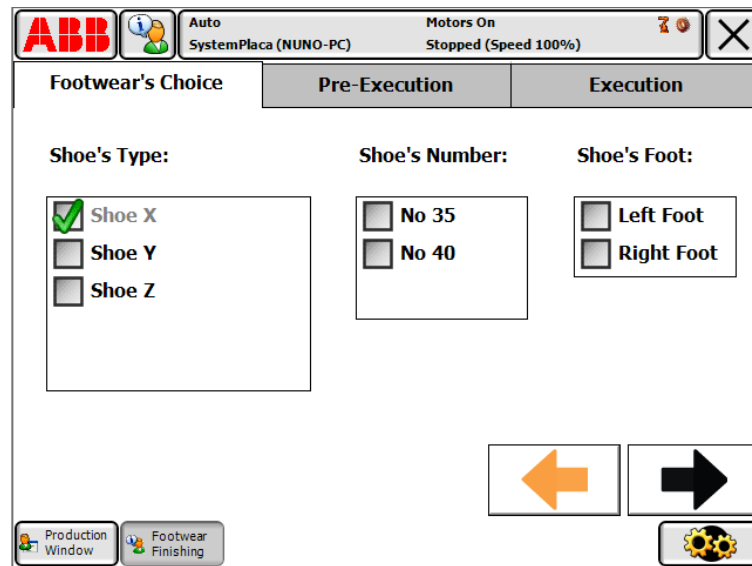
Figura 15 – Apresentação inicial do submenu *Process Shoes*

5.2.1 *Tab Footwear's Choice*

Nesta *tab* o utilizador deverá escolher o tipo, o número e orientação do sapato. Assim que as três opções se encontrem seleccionadas (Figura 16), o utilizador poderá avançar para a próxima *tab.*
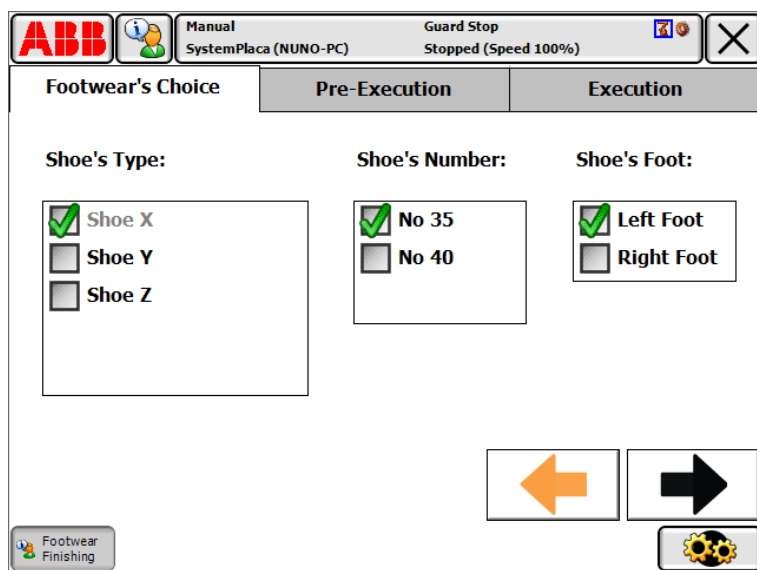
Figura 16 - Apresentação da *tab Footwear's Choice* com todas as escolhas efectuadas

Caso a selecção das opções não esteja totalmente preenchida, será exibida uma mensagem de aviso a alertar tal facto (Figura 17).
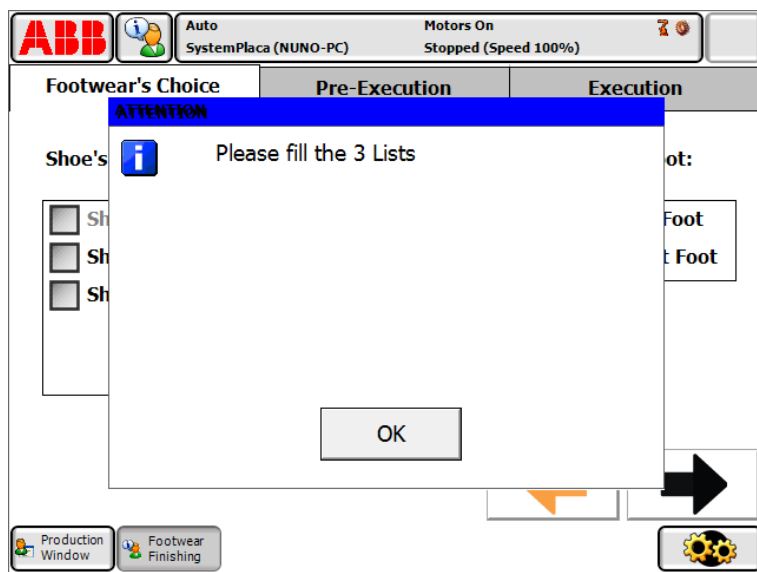
Figura 17 – Mensagem de texto a alertar a obrigatoriedade do preenchimento total das opções

5.2.2 *Tab Pre-Execution*

Nesta *tab* (Figura 18) o utilizador poderá confirmar as opções efectuadas na primeira *tab*.
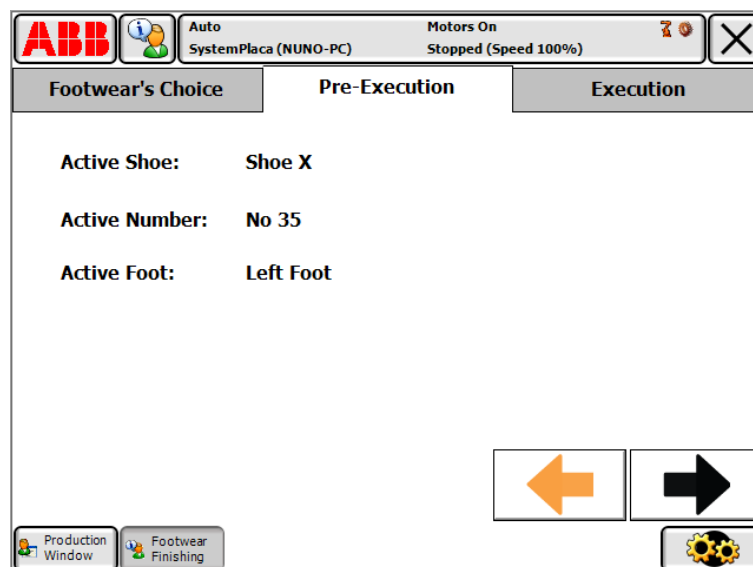
Figura 18 - Apresentação da *tab Pre-Execution*

Ao avançar para a próxima *tab*, será exibida uma mensagem de texto para reconfirmar as suas opções escolhidas (Figura 19). Caso a confirmação seja afirmativa, será apresentada a terceira *tab*. Caso contrário, a aplicação manter-se-á na mesma *tab*, dispondo da possibilidade de retroceder à *tab* anterior para modificar as opções efectuadas.
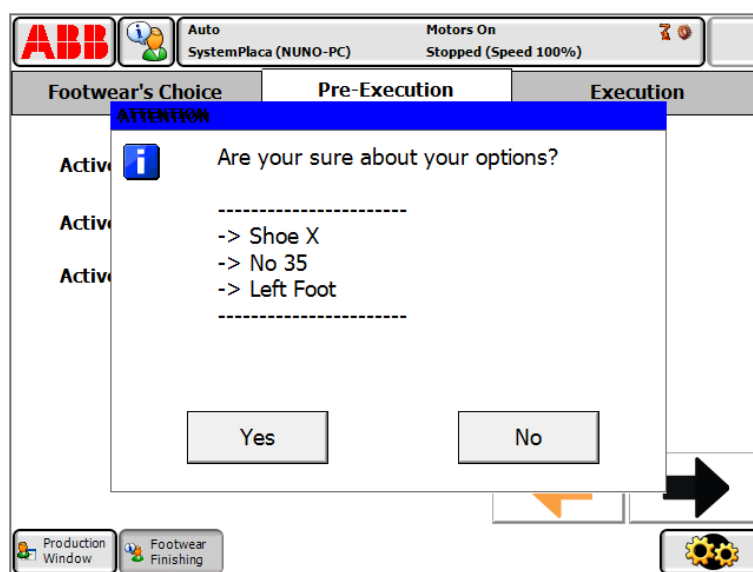
Figura 19 - Mensagem de texto a pedir a reconfirmação das escolhas efectuadas

5.2.3 *Tab Execution*

Neste *tab* (Figura 20) o utilizador terá à sua disposição três botões com as seguintes funções:

- *Start;*
- *Rotate;*
- *Stop.*
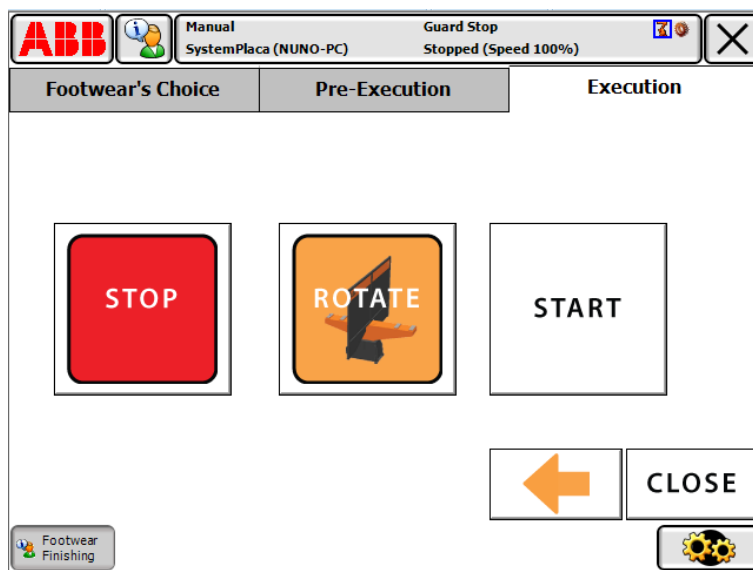
Figura 20 – Apresentação da *tab Execution*

O clique sobre o botão *Start* faz com que se inicie o processo de corte para o sapato previamente escolhido, caso a resposta à confirmação de execução seja positiva (Figura 21).
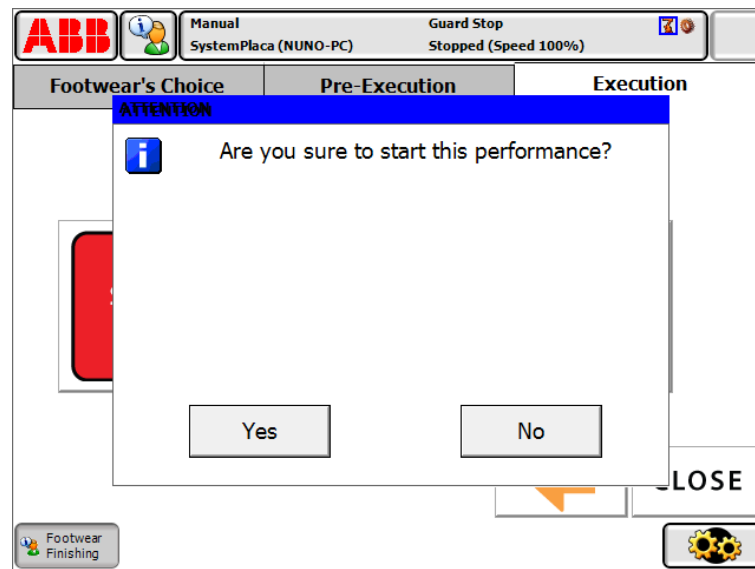
Figura 21 – Mensagem de texto pedindo confirmação de arranque do processo de corte

O clique sobre o botão *Rotate* faz com que a mesa rode cento e oitenta graus, permitindo que um novo sapato possa ser processado. Este botão apenas se torna activo depois de se ter realizado o processo de corte, o qual terminará numa posição de referência, de modo a permitir que a mesa possa rodar em segurança. Após rotação, este botão ficará de novo inactivo. De modo análogo ao procedimento aquando do clique no botão *Start*, o utilizador será questionado sobre a sua certeza relativa ao procedimento de movimentação da mesa.

O clique sobre o botão *Stop* faz com que seja efectuada uma paragem imediata em qualquer processo activo, sem que seja efectuada qualquer pergunta**.**

De forma resumida, o objectivo será o operador realizar para uma série sucessiva do mesmo sapato, a seguinte metodologia (Figura 22).
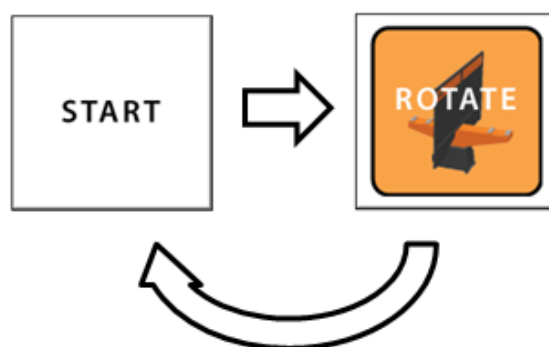
Figura 22 – Metodologia a adoptar para acabamento de uma série do mesmo sapato

Nota: Devido a falhas existentes na estrutura de apoio à programação do *Application Builder 5.1X* face ao uso de leitura de eixos externos, torna-se necessário activar a unidade mecânica da mesa (Figura 23) antes de clicar no botão rodar (Erro reportado e confirmado no fórum da *ABB*).
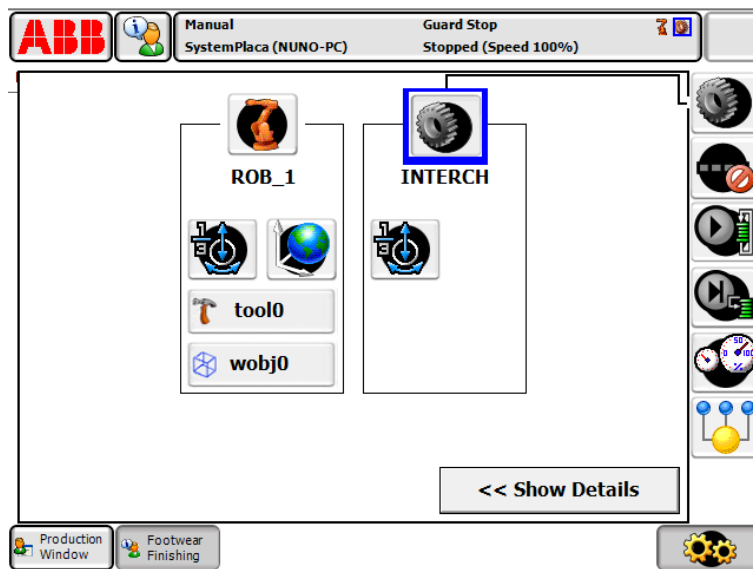
Figura 23 – Activação da unidade mecânica *INTERCH* (mesa rotativa)

5.3 Submenu *Load/Erase Shoes*

O submenu *Load/Erase Shoes* apenas poderá ser acedido por um utilizador com conhecimento da palavra-chave de acesso (Figura 24).
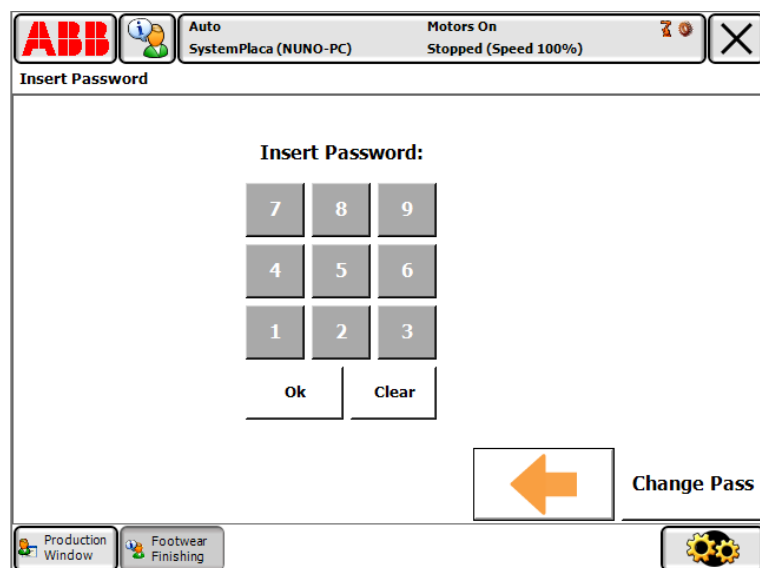
Figura 24 – Apresentação do submenu para introdução da palavra-chave

Após introdução da devida palavra-chave e clique sobre o botão *Ok*, será exibida uma mensagem de texto confirmando ou não a validade da mesma (Figura 25 e Figura 26).
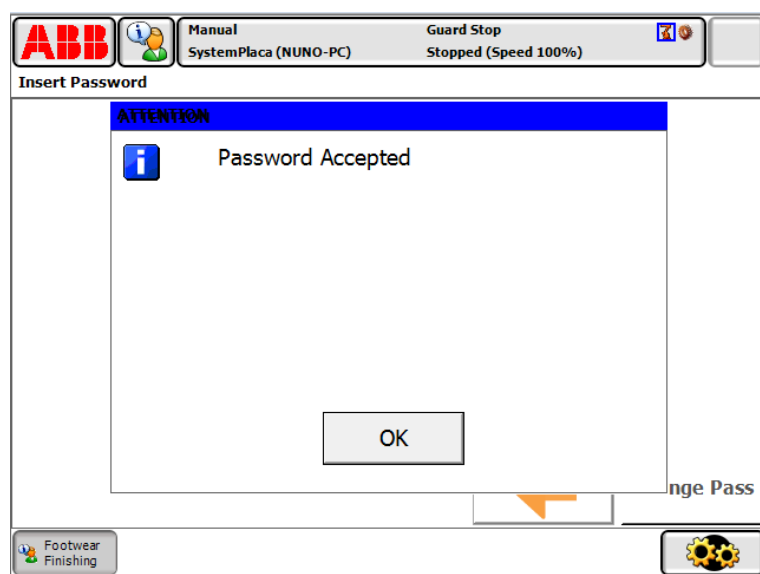
Figura 25 – Mensagem de texto a confirmar a veracidade da palavra-chave introduzida
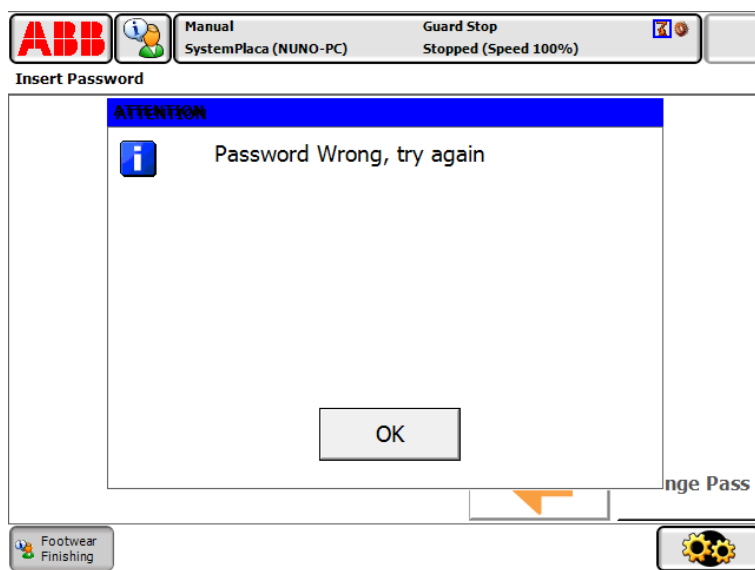
Figura 26 - Mensagem de texto a alertar o erro da palavra-chave introduzida

Caso a palavra-chave esteja correcta, será exibido o submenu *Load/Erase Shoes* (Figura 27). Caso contrário, será exibido o menu principal.
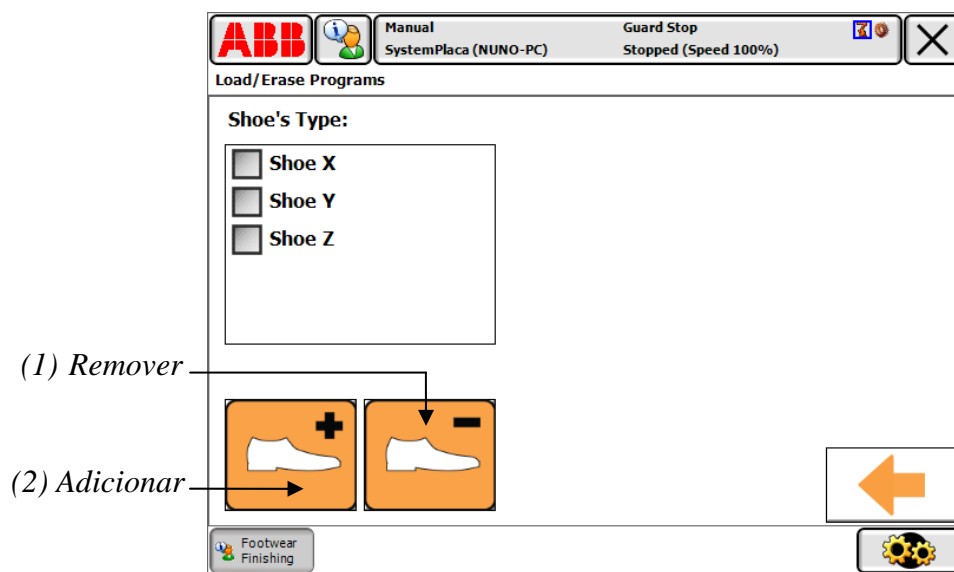
Figura 27 – Apresentação do submenu *Load/Erase Shoes*

Este submenu é constituído por dois botões que permitem adicionar ou remover sapatos da base de dados.

5.3.1 Adicionar Sapatos

Para adicionar um novo sapato à base de dados, deverá efectuar-se um clique no botão dois presente na Figura 27. Posteriormente será exibido um teclado alfanumérico (Figura 28) onde se poderá proceder à inserção do nome do sapato a adicionar. No entanto, esta operação não é suficiente para que tudo fique operacional. É necessário que exista uma pasta preparada com toda a informação do modelo inserido, de acordo com a Figura 29. Assim, ao ser adicionado um novo modelo de sapato pela primeira vez, terá de se realizar uma cópia da estrutura da pasta previamente preparada para a pasta *Shoes* (ver Figura 1). Deste modo, torna-se disponível na base de dados o novo modelo com a totalidade da sua informação.

Nota: A preparação deste conjunto de informação deverá ser realizada de modo análogo ao apresentado no tópico 3.1.
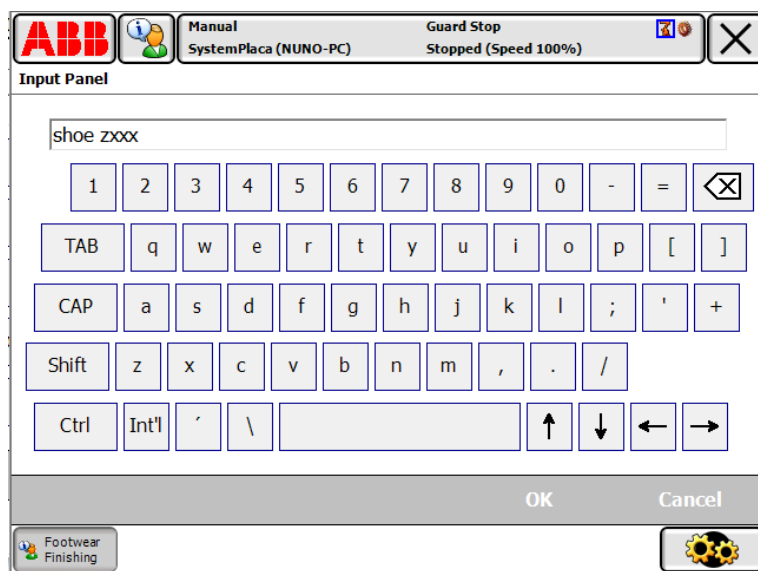
Figura 28 – Teclado alfanumérico de denominação de sapatos
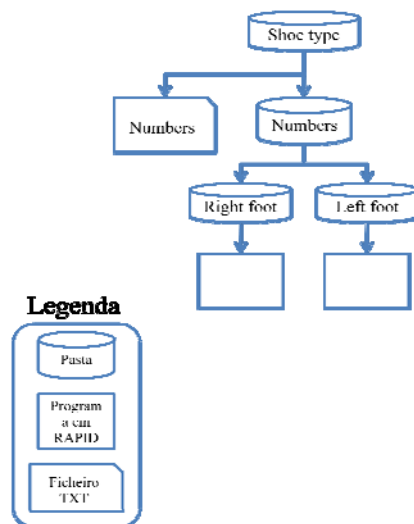
Figura 29 – Estrutura base de um modelo de sapato

5.3.2 Remover Sapatos

A remoção de sapatos da base de dados deverá ser feita seleccionando o sapato pretendido, seguido do clique do botão número um (Figura 27). Posteriormente, será exibida uma mensagem de texto de confirmação da eliminação do respectivo sapato da base de dados (Figura 30). A total eliminação dos ficheiros do sapato pretendido deverá ser feita apagando a pasta do modelo do sapato e respectivo conteúdo.
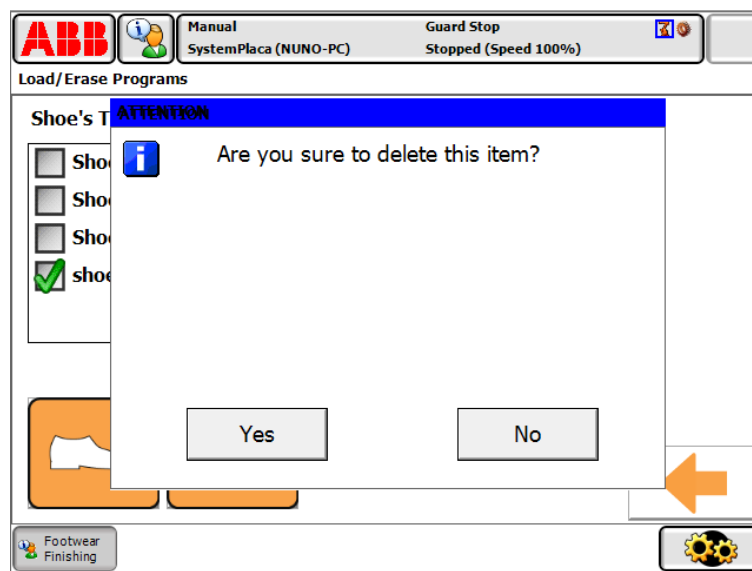
Figura 30 – Mensagem de texto de confirmação de eliminação do sapato seleccionado

5.3.3 Mudar Palavra-chave

A mudança de palavra-chave apenas poderá ser realizada por um utilizador conhecedor da palavra-chave. Este terá de efectuar os seguintes passos para a mudar:

- Clicar no botão *Change Pass*. Será exibida uma mensagem de texto pedindo que se insira a palavra-chave actual (Figura 31);
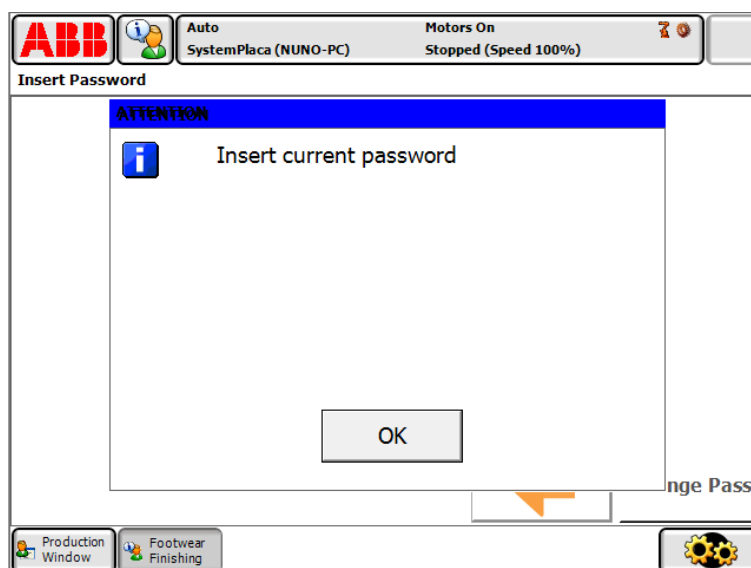
Figura 31 – Mensagem de texto para inserção de palavra-chave actual

- Clicar *Ok* presente na mensagem de texto. Introduzir a palavra-chave actual (Figura 32) e clicar novamente *Ok*.
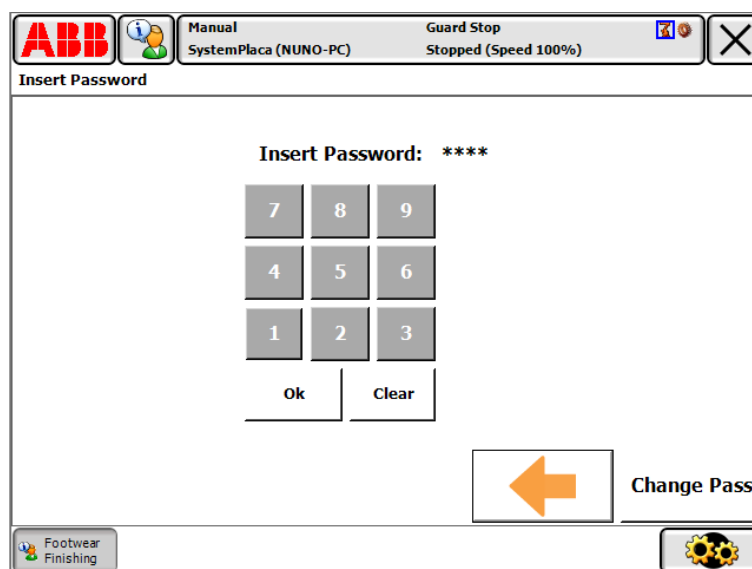
Figura 32 – Apresentação do submenu para introdução da palavra-chave

- Clicar *Ok* após aparecimento da mensagem de aceitação da palavra-chave actual. Caso a palavra-chave actual esteja correcta[1], introduzir a palavra-chave nova e clicar *Ok* presente na mensagem de texto (Figura 33).

Figura 33 - Mensagem de texto aceitando a palavra-chave actual, e a pedir a inserção de uma nova

- Após mudada a palavra-chave para uma nova, a aplicação recuará até ao menu principal, após clicar *Ok* presente na mensagem de texto (Figura 34).
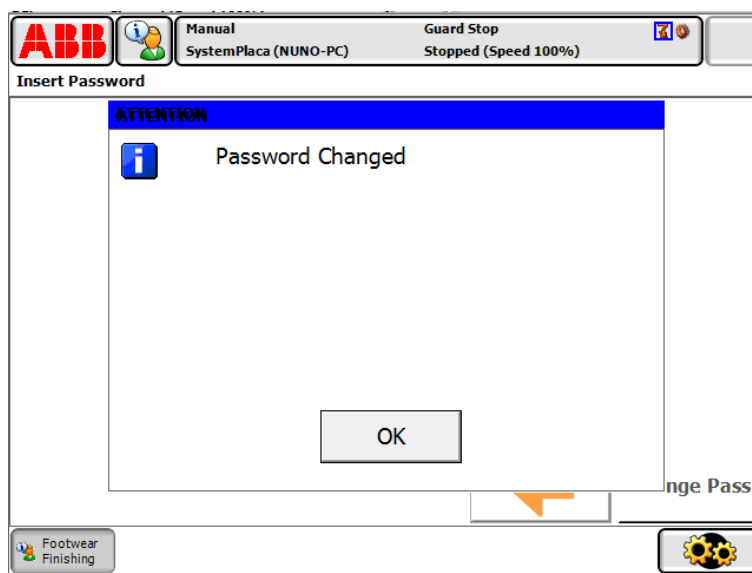
Figura 34 - Mensagem de texto confirmando a mudança de palavra-chave

[1] Caso a palavra-chave antiga esteja incorrecta, a aplicação recuará até ao menu principal, tendo que se reiniciar de novo todo o processo de mudança de palavra-chave.

5.4 Submenu *DeBug*

O submenu *DeBug* (Figura 35) fornece a possibilidade de recuperação do robô no caso de possíveis falhas ocorridas durante a execução dos programas de acabamento. As funcionalidades disponíveis neste submenu são:

- *Stop program;*
- *Deactivate Force Control;*
- *Go Home.*
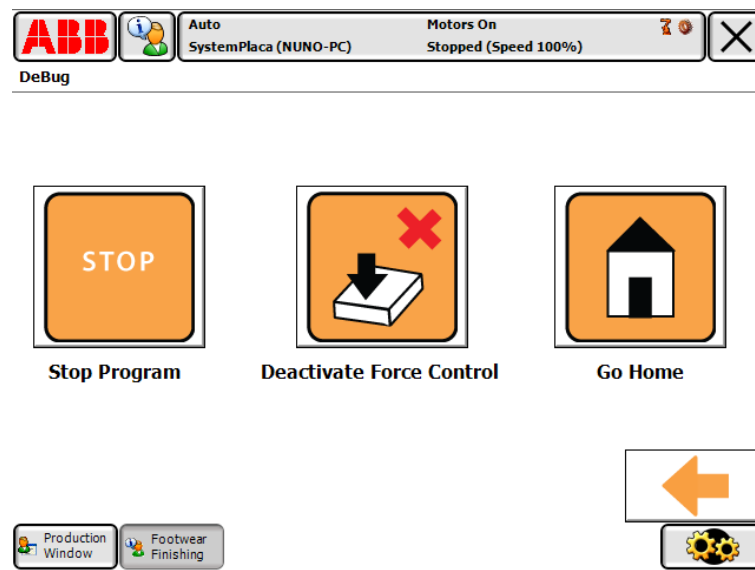
Figura 35 – Apresentação do submenu *DeBug*

Nota: O clique sobre qualquer um dos botões irá questionar sobre a intenção de iniciação da função pretendida (Figura 36).
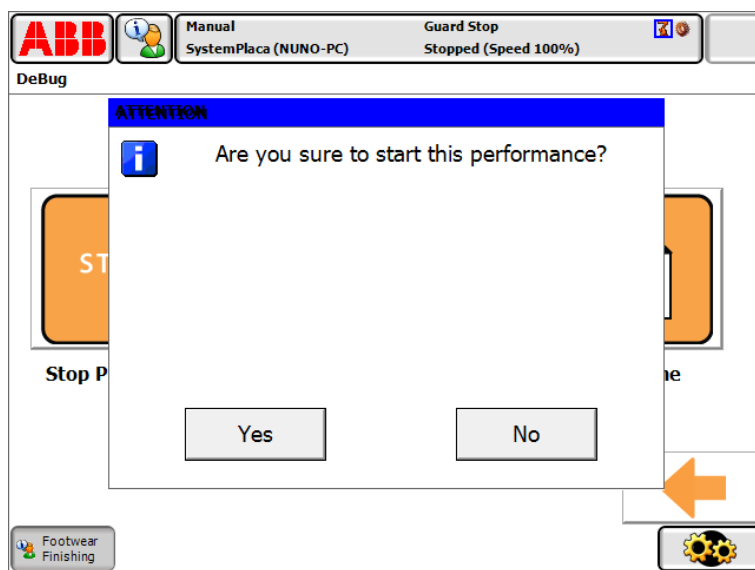
Figura 36 - Mensagem de texto pedindo confirmação de realização da função pretendida

5.4.1 *Stop Program*

O clique sobre o botão *Stop Program* efectua uma paragem forçada sobre qualquer programa a decorrer no robô. Este não se trata propriamente de um botão de emergência, uma vez que será exibida uma mensagem de texto ao utilizador sobre o facto de querer efectuar esta paragem de um programa.

5.4.2 *Deactivate Force Control*

O clique sobre o botão *Deactivate Force Control* faz com que o controlo de força seja desactivado. O utilizador deverá clicar sobre este botão sempre que seja necessário efectuar a movimentação do robô pelo uso da consola. Caso não o faça, poderão surgir oscilações não controladas na movimentação do robô.

5.4.3 *Go Home*

O clique sobre o botão *Go Home* faz com que o robô se movimente até uma posição de referência. A movimentação, efectuada até se obter esta configuração, é realizada sobre dois movimentos distintos:

- O primeiro movimento consiste na deslocação do robô no sentido contrário e perpendicular à superfície de trabalho. O afastamento relativo a esta superfície foi definido em sessenta milímetros.
- O segundo movimento consiste na deslocação do robô até à posição de referência. Esta posição encontra-se definida pelas coordenadas [0,0,0,0,30,0], expressas relativamente aos eixos das juntas do robô em graus.

O procedimento apresentado (ilustrado na Figura 37) pretende garantir que o robô não colida com o sapato. Assim, é de esperar que o robô, após ter-se deslocado os sessenta milímetros no sentido descrito, possa efectuar uma movimentação até à posição de referência sem embater no sapato.
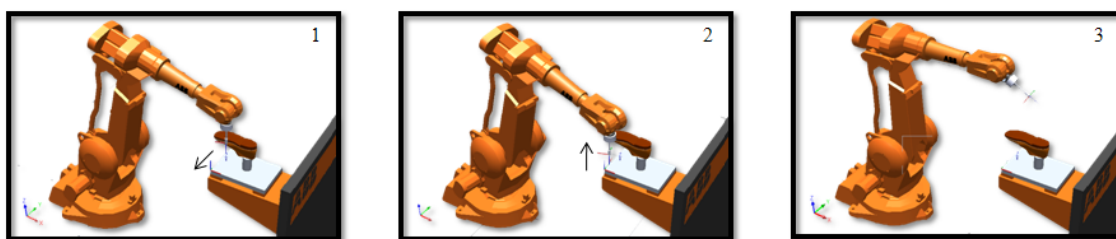
Figura 37 – Sequência de movimentação do robô até à posição de referência

Nota: Previamente à deslocação até “casa”, é efectuada automaticamente a desactivação do uso de controlo de força.

ANEXO B – Código da Aplicação Desenvolvida

1. Código do menu principal

```
using System;
using System.Drawing;
using System.Collections;

// ABB namespaces
using ABB.Robotics.Controllers;
using ABB.Robotics.Tps.Taf;
using ABB.Robotics.Tps.Windows.Forms;

// Compliance Tool Info
[assembly: TpsView("Footwear Finishing", "tpu-Operator32.gif", "tpu-
Operator16.gif", "TpsViewFootWearFinishing.dll",
"TpsViewFootWearFinishing.TpsViewFootWearFinishing",
StartPanelLocation.Left, TpsViewType.Static, TpsViewStartupTypes.Manual)]
namespace TpsViewFootWearFinishing
{
    /// <summary>
    /// Resumo descritivo do TpsViewIRC5App1.
    /// </summary>
    public class TpsViewFootWearFinishing : TpsForm, ITpsViewSetup,
ITpsViewActivation
    {
        private Button button1;

        public TpsViewFootWearFinishing()
        {
            //
            // Necessário para o windows form designer support
            //
            InitializeComponent();

            //
            // ToDo: Adicionar qualquer código contructor depois do
InitializeComponent();
            //
        }

        /// <summary>
        /// Aqui poderá ser limpo qualquer fonte utilizada pela aplicação
antes de esta ser disposed por ela própria
        /// pela TAF (TeachPendant Application Framework).
        /// O método é chamado pela TAF quando a aplicação é fechada.
        ///
        ///

        /// </summary>
        protected override void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                try
                {
```

```
                    if (disposing)
                    {
                        // ToDo: Chama o método dispose para todas as
aplicações FP SDK, evitando perdas de memória
                    }
                }
                finally
                {
                    base.Dispose(disposing);
                }
            }
        }

        #region Windows Form Designer generated code
        /// <summary>
        /// Método necessário para o designer support
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(TpsViewFootWearFinish
ing));
            this.button1 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button2 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button3 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button4 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button5 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.tpsLabel1 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel2 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel3 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.White;
            this.button1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button1.BackgroundImage")));
            this.button1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button1.Location = new System.Drawing.Point(16, 114);
            this.button1.Size = new System.Drawing.Size(168, 139);
            this.button1.TabIndex = 2;
            this.button1.Text = "";
            this.button1.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button1.Visible = false;
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.White;
            this.button2.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button2.BackgroundImage")));
            this.button2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button2.Location = new System.Drawing.Point(450, 114);
            this.button2.Size = new System.Drawing.Size(168, 139);
            this.button2.TabIndex = 3;
            this.button2.Text = "";
            this.button2.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
```

```
            this.button2.Visible = false;
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.BackColor = System.Drawing.Color.White;
            this.button3.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button3.BackgroundImage")));
            this.button3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button3.Location = new System.Drawing.Point(235, 114);
            this.button3.Size = new System.Drawing.Size(168, 139);
            this.button3.TabIndex = 4;
            this.button3.Text = "";
            this.button3.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button3.Visible = false;
            this.button3.Click += new
System.EventHandler(this.button3_Click);
            //
            // button4
            //
            this.button4.BackColor = System.Drawing.Color.Transparent;
            this.button4.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button4.BackgroundImage")));
            this.button4.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button4.Location = new System.Drawing.Point(436, 345);
            this.button4.Size = new System.Drawing.Size(91, 42);
            this.button4.TabIndex = 5;
            this.button4.Text = "";
            this.button4.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button4.Visible = false;
            this.button4.Click += new
System.EventHandler(this.button4_Click_1);
            //
            // button5
            //
            this.button5.BackColor = System.Drawing.Color.White;
            this.button5.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button5.BackgroundImage")));
            this.button5.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button5.Location = new System.Drawing.Point(533, 345);
            this.button5.Size = new System.Drawing.Size(91, 42);
            this.button5.TabIndex = 6;
            this.button5.Text = "";
            this.button5.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button5.Visible = false;
            this.button5.Click += new
System.EventHandler(this.button5_Click);
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(0, 0);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(640, 390);
```

```
            this.pictureBox1.Click += new
System.EventHandler(this.pictureBox1_Click);
            //
            // tpsLabel1
            //
            this.tpsLabel1.BackColor = System.Drawing.Color.White;
            this.tpsLabel1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel1.Location = new System.Drawing.Point(38, 259);
            this.tpsLabel1.Size = new System.Drawing.Size(146, 29);
            this.tpsLabel1.TabIndex = 7;
            this.tpsLabel1.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel1.Title = "Process Shoes";
            this.tpsLabel1.Visible = false;
            //
            // tpsLabel2
            //
            this.tpsLabel2.BackColor = System.Drawing.Color.White;
            this.tpsLabel2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel2.Location = new System.Drawing.Point(245, 259);
            this.tpsLabel2.Size = new System.Drawing.Size(195, 29);
            this.tpsLabel2.TabIndex = 8;
            this.tpsLabel2.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel2.Title = "Load/Erase Shoes";
            this.tpsLabel2.Visible = false;
            //
            // tpsLabel3
            //
            this.tpsLabel3.BackColor = System.Drawing.Color.White;
            this.tpsLabel3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel3.Location = new System.Drawing.Point(500, 259);
            this.tpsLabel3.Size = new System.Drawing.Size(84, 29);
            this.tpsLabel3.TabIndex = 9;
            this.tpsLabel3.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel3.Title = "DeBug";
            this.tpsLabel3.Visible = false;
            //
            // TpsViewFootWearFinishing
            //
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Inherit;
            this.BackColor = System.Drawing.Color.White;
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.tpsLabel3);
            this.Controls.Add(this.tpsLabel2);
            this.Controls.Add(this.tpsLabel1);
            this.Controls.Add(this.button5);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Size = new System.Drawing.Size(640, 390);
            this.Text = "Footwear Finishing";
            this.Controls.SetChildIndex(this.button1, 0);
            this.Controls.SetChildIndex(this.button2, 0);
            this.Controls.SetChildIndex(this.button3, 0);
            this.Controls.SetChildIndex(this.button4, 0);
            this.Controls.SetChildIndex(this.button5, 0);
```

```
            this.Controls.SetChildIndex(this.tpsLabel1, 0);
            this.Controls.SetChildIndex(this.tpsLabel2, 0);
            this.Controls.SetChildIndex(this.tpsLabel3, 0);
            this.Controls.SetChildIndex(this.pictureBox1, 0);
            this.ResumeLayout(false);

        }
        #endregion

        #region ITpsViewSetup Members

        /// <summary>
        /// This method is called by TAF when the control is closed down
(before Dispose is called).
        /// </summary>
        void ITpsViewSetup.Uninstall()
        {
            // TODO:  Add TpsViewFootWearFinishing.Uninstall implementation
        }

        /// <summary>
        /// This method is called by TAF when the control is installed in
the framework (after the constructor is called).
        /// </summary>
        bool ITpsViewSetup.Install(object sender, object data)
        {
            // TODO:  Add TpsViewFootWearFinishing.Install implementation
            return false;
        }

        #endregion

        #region ITpsViewActivation Members

        /// <summary>
        /// This method is called by TAF when the control goes from the
active state to the passive state,
        /// and is no longer visible in the client view. This happens when
the user presses another application button
        /// on the task bar, or closes the application. Normally, any
subscriptions to controller events are removed here.
        /// </summary>
        void ITpsViewActivation.Deactivate()
        {
            // TODO:  Add TpsViewFootWearFinishing.Deactivate
implementation
        }

        /// <summary>
        /// This method is called by TAF when the control goes from the
passive state to the active state,
        /// i.e. becomes visible in the client view. Normally, this is
where subscriptions to controller events are set up.
        /// </summary>
        void ITpsViewActivation.Activate()
        {
            // TODO:  Add TpsViewFootWearFinishing.Activate implementation
        }

        #endregion

        private Button button2;
        private view2 _view2;
```

```
        private Button button3;
        private view3 _view3;
        private Button button4;
        private Button button5;
        private System.Windows.Forms.PictureBox pictureBox1;
        private TpsLabel tpsLabel1;
        private TpsLabel tpsLabel2;
        private TpsLabel tpsLabel3;
        private view6 _view6;
        private view4 _view4;

        // Ao click, abrir novo menu -> Menu Process Shoes
        private void button1_Click(object sender, EventArgs e)
        {
            this._view2 = new view2();
            this._view2.Closed += new EventHandler(view2_Closed);
            this._view2.ShowMe(this);
        }

        //dispose da view quando esta é fechada
        void view2_Closed(object sender, EventArgs e)
        {
            this._view2.Closed -= new EventHandler(view2_Closed);
            this._view2.Dispose();
            this._view2 = null;
        }

        // Ao click, abrir novo menu -> Menu Debug
        private void button2_Click(object sender, EventArgs e)
        {
            this._view3 = new view3();
            this._view3.Closed += new EventHandler(view3_Closed);
            this._view3.ShowMe(this);
        }

        //dispose da view quando esta é fechada
        void view3_Closed(object sender, EventArgs e)
        {
            this._view3.Closed -= new EventHandler(view3_Closed);
            this._view3.Dispose();
            this._view3 = null;
        }


        private void OnServerMessageClosed_1(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
        }

        // Ao click, abrir novo menu -> Menu Password do Load/Erase Shoes

        private void button3_Click(object sender, EventArgs e)
        {
            this._view6 = new view6();
            this._view6.Closed += new EventHandler(view6_Closed);
            this._view6.ShowMe(this);
        }
        //dispose da view quando esta é fechada

        void view6_Closed(object sender, EventArgs e)
        {
            this._view6.Closed -= new EventHandler(view6_Closed);
            this._view6.Dispose();
```

```
            this._view6 = null;
        }

        //Abrir dialog box para apresentar o About !

        private void button4_Click_1(object sender, EventArgs e)
        {
            string message = "\n" + "Footwear Finishing - Version 1.0" +
"\n" + "\n" + "This program was developed only for didactic use." + "\n" +
"\n" + "\n" + "Copyright (C) by Nuno Moita @2009" + "\n" + "FEUP, MIEM";
            string caption = "About";
            GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_1), message, caption,
System.Windows.Forms.MessageBoxIcon.Exclamation,
System.Windows.Forms.MessageBoxButtons.OK);
        }

        // Condições para permitir que após click na imagem da
        // apresentação seja iniciados os objectos presentes na view
principal.

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            pictureBox1.Dispose();
            button1.Visible = true;
            button2.Visible = true;
            button3.Visible = true;
            button4.Visible = true;
            button5.Visible = true;
            tpsLabel1.Visible = true;
            tpsLabel2.Visible = true;
            tpsLabel3.Visible = true;
        }

        // Ao click, abrir novo menu -> Menu Help!

        private void button5_Click(object sender, EventArgs e)
        {
            this._view4 = new view4();
            this._view4.Closed += new EventHandler(view4_Closed);
            this._view4.ShowMe(this);
        }

        //dispose da view quando esta é fechada
        void view4_Closed(object sender, EventArgs e)
        {
            this._view4.Closed -= new EventHandler(view4_Closed);
            this._view4.Dispose();
            this._view4 = null;
        }


    }
}
```

2. Código do submenu *Process Shoes*

```
using System;
using ABB.Robotics.Controllers.RapidDomain;
using ABB.Robotics;
```

```
using ABB.Robotics.Controllers;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ABB.Robotics.Tps.Windows.Forms;
using System.IO;
using ABB.Robotics.Controllers.MotionDomain;

namespace TpsViewFootWearFinishing
{
    public class view2 : TpsForm
    {
        Controller c = new Controller();
        Task tRob1 = null;
        public int tipoSapato;
        public int tipoSapato2;
        public int tipoSapato3;
        public int x = 0;
        public int tipoSapato_1;
        public int valor;
        public float ex;

        private ABB.Robotics.Tps.Windows.Forms.TabControl tabControl1;
        private ABB.Robotics.Tps.Windows.Forms.TabPage tabPage1;
        private ABB.Robotics.Tps.Windows.Forms.TabPage tabPage2;
        private ABB.Robotics.Tps.Windows.Forms.ListView listView3;
        private ABB.Robotics.Tps.Windows.Forms.ListView listView2;
        private ABB.Robotics.Tps.Windows.Forms.ListView listView1;
        private ABB.Robotics.Tps.Windows.Forms.ListViewItem listViewItem7;
        private ABB.Robotics.Tps.Windows.Forms.ListViewItem listViewItem8;
        private TpsLabel tpsLabel4;
        private TpsLabel tpsLabel5;
        private TpsLabel tpsLabel6;
        private TpsLabel tpsLabel3;
        private TpsLabel tpsLabel2;
        private TpsLabel tpsLabel1;
        private ABB.Robotics.Tps.Windows.Forms.Button button3;
        private ABB.Robotics.Tps.Windows.Forms.Button button2;
        private ABB.Robotics.Tps.Windows.Forms.Button button4;
        private ABB.Robotics.Tps.Windows.Forms.Button button5;
        private TpsLabel tpsLabel9;
        private TpsLabel tpsLabel8;
        private TpsLabel tpsLabel7;
        private ABB.Robotics.Tps.Windows.Forms.TabPage tabPage3;
        private ABB.Robotics.Tps.Windows.Forms.Button button7;
        private ABB.Robotics.Tps.Windows.Forms.Button button6;
        private ABB.Robotics.Tps.Windows.Forms.Button button1;
        private ABB.Robotics.Tps.Windows.Forms.Button button8;
        private ABB.Robotics.Tps.Windows.Forms.Button button9;


        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;


        public view2()
        {
            InitializeComponent();
        }
```

```
        /// <summary>
        /// Clean up any resources used by this class.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                try
                {
                    if (disposing)
                    {
                        //ToDo: Call the Dispose method of all FP SDK
instances that may otherwise cause memory leak

                        if (components != null)
                        {
                            components.Dispose();
                        }
                    }
                }
                finally
                {
                    base.Dispose(disposing);
                }
            }
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(view2));
            this.tabControl1 = new
ABB.Robotics.Tps.Windows.Forms.TabControl();
            this.tabPage3 = new ABB.Robotics.Tps.Windows.Forms.TabPage();
            this.button9 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button8 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button7 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button6 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button1 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.tabPage2 = new ABB.Robotics.Tps.Windows.Forms.TabPage();
            this.button4 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button5 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.tpsLabel4 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel5 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel6 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel3 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel2 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel1 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tabPage1 = new ABB.Robotics.Tps.Windows.Forms.TabPage();
            this.tpsLabel9 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel8 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel7 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.button3 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button2 = new ABB.Robotics.Tps.Windows.Forms.Button();
```

```
            this.listView3 = new ABB.Robotics.Tps.Windows.Forms.ListView();
            this.listViewItem7 = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem();
            this.listViewItem8 = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem();
            this.listView2 = new ABB.Robotics.Tps.Windows.Forms.ListView();
            this.listView1 = new ABB.Robotics.Tps.Windows.Forms.ListView();
            this.tabControl1.SuspendLayout();
            this.tabPage3.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.SuspendLayout();
            //
            // tabControl1
            //
            this.tabControl1.BackColor = System.Drawing.Color.White;
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage3);
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tabControl1.ImageList = null;
            this.tabControl1.Location = new System.Drawing.Point(0, 0);
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(640, 390);
            this.tabControl1.TabIndex = 2;
            this.tabControl1.TabPages.Add(this.tabPage1);
            this.tabControl1.TabPages.Add(this.tabPage2);
            this.tabControl1.TabPages.Add(this.tabPage3);
            this.tabControl1.SelectedIndexChanged += new
System.EventHandler(this.tabControl1_SelectedIndexChanged);
            //
            // tabPage3
            //
            this.tabPage3.BackColor = System.Drawing.Color.White;
            this.tabPage3.Controls.Add(this.button9);
            this.tabPage3.Controls.Add(this.button8);
            this.tabPage3.Controls.Add(this.button7);
            this.tabPage3.Controls.Add(this.button6);
            this.tabPage3.Controls.Add(this.button1);
            this.tabPage3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tabPage3.ImageIndex = -1;
            this.tabPage3.Location = new System.Drawing.Point(0, 40);
            this.tabPage3.Size = new System.Drawing.Size(640, 350);
            this.tabPage3.TabIndex = 3;
            this.tabPage3.Text = "Execution";
            //
            // button9
            //
            this.button9.BackColor = System.Drawing.Color.White;
            this.button9.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button9.BackgroundImage")));
            this.button9.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button9.Location = new System.Drawing.Point(524, 286);
            this.button9.Size = new System.Drawing.Size(113, 61);
            this.button9.TabIndex = 15;
            this.button9.Text = "";
            this.button9.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button9.Click += new
System.EventHandler(this.button9_Click);
```

```
            //
            // button8
            //
            this.button8.BackColor = System.Drawing.Color.White;
            this.button8.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button8.BackgroundImage")));
            this.button8.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button8.Location = new System.Drawing.Point(408, 286);
            this.button8.Size = new System.Drawing.Size(113, 61);
            this.button8.TabIndex = 14;
            this.button8.Text = "";
            this.button8.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button8.Click += new
System.EventHandler(this.button8_Click);
            //
            // button7
            //
            this.button7.BackColor = System.Drawing.Color.White;
            this.button7.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button7.BackgroundImage")));
            this.button7.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button7.ForeColor = System.Drawing.Color.White;
            this.button7.Location = new System.Drawing.Point(408, 94);
            this.button7.Size = new System.Drawing.Size(151, 147);
            this.button7.TabIndex = 13;
            this.button7.Text = "";
            this.button7.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button7.Click += new
System.EventHandler(this.button7_Click);
            //
            // button6
            //
            this.button6.BackColor = System.Drawing.Color.White;
            this.button6.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button6.BackgroundImage")));
            this.button6.Enabled = false;
            this.button6.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button6.Location = new System.Drawing.Point(229, 94);
            this.button6.Size = new System.Drawing.Size(151, 147);
            this.button6.TabIndex = 12;
            this.button6.Text = "";
            this.button6.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button6.Click += new
System.EventHandler(this.button6_Click);
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.White;
            this.button1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button1.BackgroundImage")));
            this.button1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button1.Location = new System.Drawing.Point(38, 94);
            this.button1.Size = new System.Drawing.Size(151, 147);
            this.button1.TabIndex = 11;
            this.button1.Text = "";
```

```
            this.button1.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button1.Click += new
System.EventHandler(this.button1_Click_1);
            //
            // tabPage2
            //
            this.tabPage2.BackColor = System.Drawing.Color.Transparent;
            this.tabPage2.Controls.Add(this.button4);
            this.tabPage2.Controls.Add(this.button5);
            this.tabPage2.Controls.Add(this.tpsLabel4);
            this.tabPage2.Controls.Add(this.tpsLabel5);
            this.tabPage2.Controls.Add(this.tpsLabel6);
            this.tabPage2.Controls.Add(this.tpsLabel3);
            this.tabPage2.Controls.Add(this.tpsLabel2);
            this.tabPage2.Controls.Add(this.tpsLabel1);
            this.tabPage2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tabPage2.ImageIndex = -1;
            this.tabPage2.Location = new System.Drawing.Point(0, 40);
            this.tabPage2.Size = new System.Drawing.Size(640, 350);
            this.tabPage2.TabIndex = 2;
            this.tabPage2.Text = "Pre-Execution";
            //
            // button4
            //
            this.button4.BackColor = System.Drawing.Color.White;
            this.button4.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button4.BackgroundImage")));
            this.button4.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button4.ForeColor = System.Drawing.Color.White;
            this.button4.Location = new System.Drawing.Point(524, 286);
            this.button4.Size = new System.Drawing.Size(113, 61);
            this.button4.TabIndex = 8;
            this.button4.Text = "";
            this.button4.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button4.Click += new
System.EventHandler(this.button4_Click);
            //
            // button5
            //
            this.button5.BackColor = System.Drawing.Color.White;
            this.button5.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button5.BackgroundImage")));
            this.button5.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button5.Location = new System.Drawing.Point(408, 286);
            this.button5.Size = new System.Drawing.Size(113, 61);
            this.button5.TabIndex = 7;
            this.button5.Text = "";
            this.button5.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button5.Click += new
System.EventHandler(this.button5_Click);
            //
            // tpsLabel4
            //
            this.tpsLabel4.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel4.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel4.Location = new System.Drawing.Point(195, 120);
```

```
            this.tpsLabel4.Size = new System.Drawing.Size(326, 33);
            this.tpsLabel4.TabIndex = 5;
            this.tpsLabel4.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel4.Title = "";
            //
            // tpsLabel5
            //
            this.tpsLabel5.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel5.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel5.Location = new System.Drawing.Point(195, 75);
            this.tpsLabel5.Size = new System.Drawing.Size(326, 33);
            this.tpsLabel5.TabIndex = 4;
            this.tpsLabel5.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel5.Title = "";
            //
            // tpsLabel6
            //
            this.tpsLabel6.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel6.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel6.Location = new System.Drawing.Point(195, 26);
            this.tpsLabel6.Size = new System.Drawing.Size(326, 33);
            this.tpsLabel6.TabIndex = 3;
            this.tpsLabel6.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel6.Title = "";
            //
            // tpsLabel3
            //
            this.tpsLabel3.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel3.Location = new System.Drawing.Point(38, 120);
            this.tpsLabel3.Size = new System.Drawing.Size(129, 33);
            this.tpsLabel3.TabIndex = 2;
            this.tpsLabel3.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel3.Title = "Active Foot:";
            //
            // tpsLabel2
            //
            this.tpsLabel2.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel2.Location = new System.Drawing.Point(38, 75);
            this.tpsLabel2.Size = new System.Drawing.Size(151, 33);
            this.tpsLabel2.TabIndex = 1;
            this.tpsLabel2.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel2.Title = "Active Number:";
            //
            // tpsLabel1
            //
            this.tpsLabel1.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel1.Location = new System.Drawing.Point(38, 26);
            this.tpsLabel1.Size = new System.Drawing.Size(129, 33);
            this.tpsLabel1.TabIndex = 0;
```

```
            this.tpsLabel1.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel1.Title = "Active Shoe:";
            //
            // tabPage1
            //
            this.tabPage1.BackColor = System.Drawing.Color.White;
            this.tabPage1.Controls.Add(this.tpsLabel9);
            this.tabPage1.Controls.Add(this.tpsLabel8);
            this.tabPage1.Controls.Add(this.tpsLabel7);
            this.tabPage1.Controls.Add(this.button3);
            this.tabPage1.Controls.Add(this.button2);
            this.tabPage1.Controls.Add(this.listView3);
            this.tabPage1.Controls.Add(this.listView2);
            this.tabPage1.Controls.Add(this.listView1);
            this.tabPage1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tabPage1.ImageIndex = -1;
            this.tabPage1.Location = new System.Drawing.Point(0, 40);
            this.tabPage1.Size = new System.Drawing.Size(640, 350);
            this.tabPage1.TabIndex = 1;
            this.tabPage1.Text = "Footwear\'s Choice";
            //
            // tpsLabel9
            //
            this.tpsLabel9.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel9.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel9.Location = new System.Drawing.Point(478, 26);
            this.tpsLabel9.Size = new System.Drawing.Size(147, 29);
            this.tpsLabel9.TabIndex = 7;
            this.tpsLabel9.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel9.Title = "Shoe\'s Foot:";
            //
            // tpsLabel8
            //
            this.tpsLabel8.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel8.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel8.Location = new System.Drawing.Point(292, 26);
            this.tpsLabel8.Size = new System.Drawing.Size(147, 29);
            this.tpsLabel8.TabIndex = 6;
            this.tpsLabel8.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel8.Title = "Shoe\'s Number:";
            //
            // tpsLabel7
            //
            this.tpsLabel7.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel7.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel7.Location = new System.Drawing.Point(28, 26);
            this.tpsLabel7.Size = new System.Drawing.Size(147, 29);
            this.tpsLabel7.TabIndex = 5;
            this.tpsLabel7.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel7.Title = "Shoe\'s Type:";
            //
            // button3
            //
            this.button3.BackColor = System.Drawing.Color.White;
```

```
            this.button3.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button3.BackgroundImage")));
            this.button3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button3.Location = new System.Drawing.Point(524, 286);
            this.button3.Size = new System.Drawing.Size(113, 61);
            this.button3.TabIndex = 4;
            this.button3.Text = "";
            this.button3.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button3.Click += new
System.EventHandler(this.button3_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.White;
            this.button2.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button2.BackgroundImage")));
            this.button2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button2.Location = new System.Drawing.Point(405, 286);
            this.button2.Size = new System.Drawing.Size(113, 61);
            this.button2.TabIndex = 3;
            this.button2.Text = "";
            this.button2.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // listView3
            //
            this.listView3.BackColor = System.Drawing.Color.White;
            this.listView3.BorderSize = 1;
            this.listView3.CheckBoxes = true;
            this.listView3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.listView3.ForeColor = System.Drawing.Color.Black;
            this.listView3.HeaderStyle =
System.Windows.Forms.ColumnHeaderStyle.Clickable;
            this.listView3.ItemHeigth = -1;
            this.listView3.Items.Add(this.listViewItem7);
            this.listView3.Items.Add(this.listViewItem8);
            this.listView3.LargeImageList = null;
            this.listView3.ListViewItemSorter = null;
            this.listView3.Location = new System.Drawing.Point(478, 75);
            this.listView3.MultiSelect = false;
            this.listView3.Scrollable = true;
            this.listView3.SelectedIndex = -1;
            this.listView3.SelectionEnabledOverScrollbuttons = false;
            this.listView3.ShowNumberOfItems = false;
            this.listView3.ShowSelection = false;
            this.listView3.Size = new System.Drawing.Size(139, 69);
            this.listView3.SmallImageList = null;
            this.listView3.Sorting =
ABB.Robotics.Tps.Windows.Forms.SortOrder.Ascending;
            this.listView3.SubItemsImageList = null;
            this.listView3.TabIndex = 2;
            this.listView3.View = System.Windows.Forms.View.LargeIcon;
            this.listView3.Zoomable = false;
            this.listView3.SelectedIndexChanged += new
System.EventHandler(this.listView3_SelectedIndexChanged);
            //
            // listViewItem7
```

```
            //
            this.listViewItem7.CheckBoxVisible = true;
            this.listViewItem7.Checked = false;
            this.listViewItem7.Enabled = true;
            this.listViewItem7.ImageIndex = -1;
            this.listViewItem7.ItemName = "LVItem";
            this.listViewItem7.ListView = this.listView3;
            this.listViewItem7.Selected = false;
            this.listViewItem7.Tag = null;
            this.listViewItem7.Text = "Left Foot";
            //
            // listViewItem8
            //
            this.listViewItem8.CheckBoxVisible = true;
            this.listViewItem8.Checked = false;
            this.listViewItem8.Enabled = true;
            this.listViewItem8.ImageIndex = -1;
            this.listViewItem8.ItemName = "LVItem";
            this.listViewItem8.ListView = this.listView3;
            this.listViewItem8.Selected = false;
            this.listViewItem8.Tag = null;
            this.listViewItem8.Text = "Right Foot";
            //
            // listView2
            //
            this.listView2.BackColor = System.Drawing.Color.White;
            this.listView2.BorderSize = 1;
            this.listView2.CheckBoxes = true;
            this.listView2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.listView2.ForeColor = System.Drawing.Color.Black;
            this.listView2.HeaderStyle =
System.Windows.Forms.ColumnHeaderStyle.Clickable;
            this.listView2.ItemHeigth = -1;
            this.listView2.LargeImageList = null;
            this.listView2.ListViewItemSorter = null;
            this.listView2.Location = new System.Drawing.Point(292, 75);
            this.listView2.MultiSelect = false;
            this.listView2.Scrollable = true;
            this.listView2.SelectedIndex = -1;
            this.listView2.SelectionEnabledOverScrollbuttons = false;
            this.listView2.ShowNumberOfItems = false;
            this.listView2.ShowSelection = false;
            this.listView2.Size = new System.Drawing.Size(147, 105);
            this.listView2.SmallImageList = null;
            this.listView2.Sorting =
ABB.Robotics.Tps.Windows.Forms.SortOrder.Ascending;
            this.listView2.SubItemsImageList = null;
            this.listView2.TabIndex = 1;
            this.listView2.View = System.Windows.Forms.View.LargeIcon;
            this.listView2.Zoomable = false;
            //
            // listView1
            //
            this.listView1.BackColor = System.Drawing.Color.White;
            this.listView1.BorderSize = 1;
            this.listView1.CheckBoxes = true;
            this.listView1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.listView1.ForeColor = System.Drawing.Color.Black;
            this.listView1.HeaderStyle =
System.Windows.Forms.ColumnHeaderStyle.Clickable;
            this.listView1.ItemHeigth = -1;
```

```
            this.listView1.LargeImageList = null;
            this.listView1.ListViewItemSorter = null;
            this.listView1.Location = new System.Drawing.Point(28, 75);
            this.listView1.MultiSelect = false;
            this.listView1.Scrollable = true;
            this.listView1.SelectedIndex = -1;
            this.listView1.SelectionEnabledOverScrollbuttons = false;
            this.listView1.ShowNumberOfItems = false;
            this.listView1.ShowSelection = false;
            this.listView1.Size = new System.Drawing.Size(226, 166);
            this.listView1.SmallImageList = null;
            this.listView1.Sorting =
ABB.Robotics.Tps.Windows.Forms.SortOrder.Ascending;
            this.listView1.SubItemsImageList = null;
            this.listView1.TabIndex = 0;
            this.listView1.View = System.Windows.Forms.View.LargeIcon;
            this.listView1.Zoomable = false;
            this.listView1.Click += new
System.EventHandler(this.listView1_Click);
            //
            // view2
            //
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Inherit;
            this.BackColor = System.Drawing.Color.White;
            this.Controls.Add(this.tabControl1);
            this.Size = new System.Drawing.Size(640, 390);
            this.Text = "Shoe\'s Finishing";
            this.Load += new System.EventHandler(this.view2_Load);
            this.Controls.SetChildIndex(this.tabControl1, 0);
            this.tabControl1.ResumeLayout(false);
            this.tabPage3.ResumeLayout(false);
            this.tabPage2.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.ResumeLayout(false);

        }

        #endregion

        private void button3_Click(object sender, EventArgs e)
        {
            // Verificação se as 3 listas estão seleccionadas

            if (listView1.SelectedIndex != -1 && listView2.SelectedIndex !=
-1 && listView3.SelectedIndex != -1 && listView1.SelectedItem.Checked ==
true && listView2.SelectedItem.Checked == true &&
listView3.SelectedItem.Checked == true)
            {
                x = 1;
                tabControl1.SelectedIndex = 1;
                tipoSapato = listView1.SelectedItem.Index;
                tipoSapato2 = listView2.SelectedItem.Index;
                tipoSapato3 = listView3.SelectedItem.Index;
                tpsLabel6.Text = listView1.Items[tipoSapato].Text;
                tpsLabel5.Text = listView2.Items[tipoSapato2].Text;
                tpsLabel4.Text = listView3.Items[tipoSapato3].Text;
            }

            else
            {
                // Caso contrário deselecciona todos e index das tabs a -1.
```

```
                if (listView1.SelectedIndex != -1)
                {
                    tipoSapato = listView1.SelectedItem.Index;
                    listView1.Items[tipoSapato].Checked = false;
                    listView1.SelectedIndex = -1;

                }
                if (listView2.SelectedIndex != -1)
                {
                    tipoSapato2 = listView2.SelectedItem.Index;
                    listView2.Items[tipoSapato2].Checked = false;
                    listView2.SelectedIndex = -1;
                }
                if (listView3.SelectedIndex != -1)
                {
                    tipoSapato3 = listView3.SelectedItem.Index;
                    listView3.Items[tipoSapato3].Checked = false;
                    listView3.SelectedIndex = -1;
                }

                //Chamar Caixa de Texto de Aviso devido à não selecção das
3 listas.

                string message = "Please fill the 3 Lists";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
                listView1.Items[valor].Enabled = true;

            }
        }


        private void OnServerMessageClosed(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
        }

        private void button4_Click(object sender, EventArgs e)
       {
            //Ir buscar informação aos index seleccionados das selecções
            //e mostrar activadas as opções

                x = 2;
                tipoSapato = listView1.SelectedItem.Index;
                tipoSapato2 = listView2.SelectedItem.Index;
                tipoSapato3 = listView3.SelectedItem.Index;
                tpsLabel6.Text = listView1.Items[tipoSapato].Text;
                tpsLabel5.Text = listView2.Items[tipoSapato2].Text;
                tpsLabel4.Text = listView3.Items[tipoSapato3].Text;

             //Chamar Caixa de Texto de Aviso sobre as opções

                string message = "Are your sure about your options?";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_button_4), message + "\n" +
"\n" + "-----------------------" + "\n" + "-> " +
listView1.SelectedItem.Text + "\n" + "-> " + listView2.SelectedItem.Text +
"\n" + "-> " + listView3.SelectedItem.Text + "\n" + "----------------------
```

```
-", caption, System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);
            }


        private void OnServerMessageClosed_button_4(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            //Quando se carrega YES, vai abrir a tab Execution

            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {
                tabControl1.SelectedIndex = 2;

            }
            else if (e.DialogResult ==
System.Windows.Forms.DialogResult.No)
            {
                //Caso contrário, fica na mesma tab!

                tabControl1.SelectedIndex = 1;
            }
        }


        private void button5_Click(object sender, EventArgs e)
        {
            //Botão de retroceder, vai para a tab das selecções de sapato!

            x = 0;
            tabControl1.SelectedIndex = 0;

        }


        private void button2_Click(object sender, EventArgs e)
        {
            //Fechar a janela e ir para o menu principal, libertando a
memória
            //do controlador

            this.Dispose();
        }

        private void tabControl1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            //Implementação de código de modo a permitir bloquear as tabs
            //Os X's vão ser alterados de acordo com os botões presentes
            //em cada tab!

            if (x == 0)
            {
                tabControl1.SelectedIndex = 0;
            }
            if (x == 1)
            {
                tabControl1.SelectedIndex = 1;
            }
            if (x == 2)
            {
                tabControl1.SelectedIndex = 2;
            }
```

```
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //Paragem de emergência de qualquer programa rapid a correr!

            Rapid rapid = c.Rapid;
            rapid.Stop();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            //Fechar a janela e ir para o menu principal, libertando a
memória
            //do controlador

            this.Dispose();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            //Questão sobre a certeza do utilizador perante o inicio de um
programa

            string message = "Are you sure to start this performance?";
            string caption = "ATTENTION";
            GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_2), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);
        }

        //implementar programa de recurso ao uso de Yes ou No

        private void OnServerMessageClosed_2(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {
                //Ir ao controlador T_ROB1 e faz o load do programa da
pasta!!
                button6.Enabled = true;
                Rapid rapid = c.Rapid;
                tRob1 = c.Rapid.GetTask("T_ROB1");

                //Retira o nome de cada selecção de sapato e vai modificar
                //O caminho para ir fazer load à respectiva pasta.

                string path = "C:/FOOTWEAR/Shoes/" +
listView1.SelectedItem.Text.Replace(" ", "") + "/" +
listView2.SelectedItem.Text.Replace(" ", "") + "/" +
listView3.SelectedItem.Text.Replace(" ", "") + "/Shoe.PGF";

                tRob1.LoadProgramFromFile(path, RapidLoadMode.Replace);

                rapid.ResetProgramPointer();
                //Executar o Programa Rapid
                rapid.Start();

            }
        }

        private void button8_Click(object sender, EventArgs e)
```

```
        {
            //Botão de retroceder para o menu Execution

            x = 1;
            tabControl1.SelectedIndex = 1;
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            //Paragem de Emergência

            Rapid rapid = c.Rapid;
            tRob1 = c.Rapid.GetTask("T_ROB1");
            rapid.Stop();
        }


        private void view2_Load(object sender, EventArgs e)
        {
            //vai carregar para a variavel lista o conteudo do ficheiro
            //onde está a base dos sapatos

            string path = "C:/FOOTWEAR/Shoes/Shoes.txt";


            StreamReader list = new StreamReader(path);

            /*ciclo, que enquanto nao chegar ao final do ficheiro
            vai criar novos items com o nome da existente em cada linha do
ficheiro
            e depois adiciona-los a' listbox */

            while (!list.EndOfStream)
            {
                ABB.Robotics.Tps.Windows.Forms.ListViewItem item = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem(list.ReadLine());
                listView1.Items.Add(item);

            }
            // fecha a leitura
            list.Close();

            //Quando se abre a janela Process Shoes na parte da selecção de
sapatos
            //O programa coloca selecionado logo o primeiro elemento
            //De modo a evitar que a lista dos números não esteja em
branco;
            //O elemento selecionado é também bloqueado por questões de
simplificação de código
            //Uma vez que cada vez que se selecionava e deseleccionada o
mesmo elemento,
            //não era feito um refresh!!

            tipoSapato_1 = 0;
            listView1.Items[tipoSapato_1].Checked = true;
            listView1.SelectedIndex = tipoSapato_1;
            listView1.Items[tipoSapato_1].Enabled = false;
            valor = tipoSapato_1;

            //vai carregar para a lista de número do sapato seleccionado
```

```
            string path_1 = "C:/FOOTWEAR/Shoes/" +
listView1.Items[tipoSapato_1].Text.Replace(" ", "") + "/" +
listView1.Items[tipoSapato_1].Text.Replace(" ", "") + ".txt";

            listView2.Items.Clear();

            StreamReader list_1 = new StreamReader(path_1);

            /*ciclo, que enquanto nao chegar ao final do ficheiro
            vai criar novos items com o nome da existente em cada linha do
ficheiro
            e depois adiciona-los a' listbox */

            while (!list_1.EndOfStream)
            {
                ABB.Robotics.Tps.Windows.Forms.ListViewItem item = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem(list_1.ReadLine());
                listView2.Items.Add(item);
            }
            // fecha a leitura

            list_1.Close();

        }

        //Quando se modificar a selecção feita na parte da selecção dos
sapatos

        private void listView1_Click(object sender, EventArgs e)
        {
            //De modo a evitar que a lista dos números não esteja em
branco;
            //O elemento selecionado é também bloqueado por questões de
simplificação de código
            //Uma vez que cada vez que se selecionava e deseleccionada o
mesmo elemento,
            //não era feito um refresh!

            if (listView1.SelectedIndex != -1)
            {
                listView1.Items[valor].Enabled = true;
                tipoSapato_1 = listView1.SelectedIndex;
                listView1.Items[tipoSapato_1].Enabled = false;
                valor = tipoSapato_1;

                string path = "C:/FOOTWEAR/Shoes/" +
listView1.SelectedItem.Text.Replace(" ", "") + "/" +
listView1.SelectedItem.Text.Replace(" ", "") + ".txt";

                //vai carregar para a lista de número do sapato
seleccionado

                listView2.Items.Clear();

                StreamReader list = new StreamReader(path);

                /*ciclo, que enquanto nao chegar ao final do ficheiro
                vai criar novos items com o nome da existente em cada linha
do ficheiro
                e depois adiciona-los a' listbox */

                while (!list.EndOfStream)
                {
```

```
                    ABB.Robotics.Tps.Windows.Forms.ListViewItem item = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem(list.ReadLine());
                    listView2.Items.Add(item);

                }
                // fecha a leitura
                list.Close();

            }
        }

        //Ao clicar Rodar mesa, vai questionar a certeza do utilizador

        private void button6_Click(object sender, EventArgs e)
        {
            string message = "Are you sure about it?";
            string caption = "ATTENTION";
            GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_4), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);
        }

        private void OnServerMessageClosed_4(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            //Botão Rodar mesa

            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {
                //Vai ler o valor da posição da mesa

                MotionSystem motion = c.MotionSystem;
                MechanicalUnit mech = motion.GetActiveMechanicalUnit();
                JointTarget robb;
                robb = mech.GetPosition();
                ex = robb.RobAx.Rax_1;

                // Por segurança, vai confirmar se este se chama INTERCH
(Nome da unidade mecânica)

                if (mech.Name == "INTERCH")
                {

                    //Margem de erro possivel, para 0 graus.

                    if (ex >= -5 && ex <= 5)
                    {
                        // Vai carregar programa para mover 180

                        string path_1 =
"C:/FOOTWEAR/Applications/Rotate/Rotate180/Rotate180.pgf";
                        tRob1.LoadProgramFromFile(path_1,
RapidLoadMode.Replace);


                        //Bloquear Botão Rodar
                        button6.Enabled = false;

                        //Executar o Programa Rapid
                        Rapid rapid = c.Rapid;
                        rapid.ResetProgramPointer();
```

```
                        //iniciar programa
                        rapid.Start();



                    }

                    //Margem de erro possivel, para 180 graus.

                    if (ex >= 175 && ex <= 185)
                    {
                        // Vai carregar programa para mover 0

                        string path_2 =
"C:/FOOTWEAR/Applications/Rotate/Rotate0/Rotate0.pgf";
                        tRob1.LoadProgramFromFile(path_2,
RapidLoadMode.Replace);


                        //Bloquear Botão Rodar
                        button6.Enabled = false;
                        //Executar o Programa Rapid
                        Rapid rapid = c.Rapid;
                        rapid.ResetProgramPointer();
                        //iniciar programa

                        rapid.Start();

                    }
                }
            }
        }

        private void listView3_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }
    }
}
```

3. Código do submenu *Load/Erase Shoes*

3.1. Submenu palavra-chave

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ABB.Robotics.Tps.Windows.Forms;
using System.IO;

namespace TpsViewFootWearFinishing
{
    public class view6 : TpsForm
    {
        private view5 _view5;
```

```
        private int x = 1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Button button5;
        private System.Windows.Forms.Button button6;
        private System.Windows.Forms.Button button7;
        private System.Windows.Forms.Button button8;
        private System.Windows.Forms.Button button9;
        private System.Windows.Forms.Button button10;
        private System.Windows.Forms.Button button11;
        private ABB.Robotics.Tps.Windows.Forms.Button button12;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private Label label1;
        private System.Windows.Forms.Button button13;
        private System.Windows.Forms.Button button14;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;


        public view6()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Clean up any resources used by this class.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                try
                {
                    if (disposing)
                    {
                        //ToDo: Call the Dispose method of all FP SDK
instances that may otherwise cause memory leak

                        if (components != null)
                        {
                            components.Dispose();
                        }
                    }
                }
                finally
                {
                    base.Dispose(disposing);
                }
            }
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
```

```
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(view6));
            this.button1 = new System.Windows.Forms.Button();
            this.button2 = new System.Windows.Forms.Button();
            this.button3 = new System.Windows.Forms.Button();
            this.button4 = new System.Windows.Forms.Button();
            this.button5 = new System.Windows.Forms.Button();
            this.button6 = new System.Windows.Forms.Button();
            this.button7 = new System.Windows.Forms.Button();
            this.button8 = new System.Windows.Forms.Button();
            this.button9 = new System.Windows.Forms.Button();
            this.button10 = new System.Windows.Forms.Button();
            this.button11 = new System.Windows.Forms.Button();
            this.button12 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.button13 = new System.Windows.Forms.Button();
            this.button14 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.DarkGray;
            this.button1.ForeColor = System.Drawing.Color.White;
            this.button1.Location = new System.Drawing.Point(198, 204);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(50, 46);
            this.button1.TabIndex = 2;
            this.button1.Text = "1";
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.DarkGray;
            this.button2.ForeColor = System.Drawing.Color.White;
            this.button2.Location = new System.Drawing.Point(254, 204);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(50, 46);
            this.button2.TabIndex = 3;
            this.button2.Text = "2";
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.BackColor = System.Drawing.Color.DarkGray;
            this.button3.ForeColor = System.Drawing.Color.White;
            this.button3.Location = new System.Drawing.Point(310, 204);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(50, 46);
            this.button3.TabIndex = 4;
            this.button3.Text = "3";
            this.button3.Click += new
System.EventHandler(this.button3_Click);
            //
            // button4
            //
```

```
            this.button4.BackColor = System.Drawing.Color.DarkGray;
            this.button4.ForeColor = System.Drawing.Color.White;
            this.button4.Location = new System.Drawing.Point(198, 152);
            this.button4.Name = "button4";
            this.button4.Size = new System.Drawing.Size(50, 46);
            this.button4.TabIndex = 5;
            this.button4.Text = "4";
            this.button4.Click += new
System.EventHandler(this.button4_Click);
            //
            // button5
            //
            this.button5.BackColor = System.Drawing.Color.DarkGray;
            this.button5.ForeColor = System.Drawing.Color.White;
            this.button5.Location = new System.Drawing.Point(254, 152);
            this.button5.Name = "button5";
            this.button5.Size = new System.Drawing.Size(50, 46);
            this.button5.TabIndex = 6;
            this.button5.Text = "5";
            this.button5.Click += new
System.EventHandler(this.button5_Click);
            //
            // button6
            //
            this.button6.BackColor = System.Drawing.Color.DarkGray;
            this.button6.ForeColor = System.Drawing.Color.White;
            this.button6.Location = new System.Drawing.Point(310, 152);
            this.button6.Name = "button6";
            this.button6.Size = new System.Drawing.Size(50, 46);
            this.button6.TabIndex = 7;
            this.button6.Text = "6";
            this.button6.Click += new
System.EventHandler(this.button6_Click);
            //
            // button7
            //
            this.button7.BackColor = System.Drawing.Color.DarkGray;
            this.button7.ForeColor = System.Drawing.Color.White;
            this.button7.Location = new System.Drawing.Point(198, 100);
            this.button7.Name = "button7";
            this.button7.Size = new System.Drawing.Size(50, 46);
            this.button7.TabIndex = 8;
            this.button7.Text = "7";
            this.button7.Click += new
System.EventHandler(this.button7_Click);
            //
            // button8
            //
            this.button8.BackColor = System.Drawing.Color.DarkGray;
            this.button8.ForeColor = System.Drawing.Color.White;
            this.button8.Location = new System.Drawing.Point(254, 100);
            this.button8.Name = "button8";
            this.button8.Size = new System.Drawing.Size(50, 46);
            this.button8.TabIndex = 9;
            this.button8.Text = "8";
            this.button8.Click += new
System.EventHandler(this.button8_Click);
            //
            // button9
            //
            this.button9.BackColor = System.Drawing.Color.DarkGray;
            this.button9.ForeColor = System.Drawing.Color.White;
            this.button9.Location = new System.Drawing.Point(310, 100);
```

```
            this.button9.Name = "button9";
            this.button9.Size = new System.Drawing.Size(50, 46);
            this.button9.TabIndex = 10;
            this.button9.Text = "9";
            this.button9.Click += new
System.EventHandler(this.button9_Click);
            //
            // button10
            //
            this.button10.BackColor = System.Drawing.Color.White;
            this.button10.Font = new System.Drawing.Font("Tahoma", 10F,
System.Drawing.FontStyle.Bold);
            this.button10.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            this.button10.Location = new System.Drawing.Point(198, 256);
            this.button10.Name = "button10";
            this.button10.Size = new System.Drawing.Size(83, 45);
            this.button10.TabIndex = 11;
            this.button10.Text = "Ok";
            this.button10.Click += new
System.EventHandler(this.button10_Click);
            //
            // button11
            //
            this.button11.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.button11.BackColor = System.Drawing.Color.White;
            this.button11.Font = new System.Drawing.Font("Tahoma", 10F,
System.Drawing.FontStyle.Bold);
            this.button11.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            this.button11.Location = new System.Drawing.Point(287, 256);
            this.button11.Name = "button11";
            this.button11.Size = new System.Drawing.Size(73, 45);
            this.button11.TabIndex = 12;
            this.button11.Text = "Clear";
            this.button11.Click += new
System.EventHandler(this.button11_Click);
            //
            // button12
            //
            this.button12.BackColor = System.Drawing.Color.White;
            this.button12.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button12.BackgroundImage")));
            this.button12.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button12.Location = new System.Drawing.Point(391, 325);
            this.button12.Size = new System.Drawing.Size(120, 62);
            this.button12.TabIndex = 13;
            this.button12.Text = "";
            this.button12.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button12.Click += new
System.EventHandler(this.button12_Click);
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(365, 36);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(116, 23);
            this.textBox1.TabIndex = 14;
            this.textBox1.Visible = false;
```

```
            //
            // textBox2
            //
            this.textBox2.BorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.textBox2.Font = new System.Drawing.Font("Tahoma", 12F,
System.Drawing.FontStyle.Bold);
            this.textBox2.Location = new System.Drawing.Point(365, 65);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(116, 26);
            this.textBox2.TabIndex = 15;
            //
            // label1
            //
            this.label1.BackColor = System.Drawing.Color.White;
            this.label1.Font = new System.Drawing.Font("Tahoma", 12F,
System.Drawing.FontStyle.Bold);
            this.label1.ForeColor = System.Drawing.Color.Black;
            this.label1.Location = new System.Drawing.Point(207, 64);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(152, 24);
            this.label1.Text = "Insert Password:";
            //
            // button13
            //
            this.button13.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.button13.BackColor = System.Drawing.Color.White;
            this.button13.Font = new System.Drawing.Font("Tahoma", 12F,
System.Drawing.FontStyle.Bold);
            this.button13.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            this.button13.Location = new System.Drawing.Point(517, 325);
            this.button13.Name = "button13";
            this.button13.Size = new System.Drawing.Size(120, 62);
            this.button13.TabIndex = 16;
            this.button13.Text = "Change Pass";
            this.button13.Click += new
System.EventHandler(this.button13_Click);
            //
            // button14
            //
            this.button14.Anchor = System.Windows.Forms.AnchorStyles.None;
            this.button14.BackColor = System.Drawing.Color.White;
            this.button14.Font = new System.Drawing.Font("Tahoma", 10F,
System.Drawing.FontStyle.Bold);
            this.button14.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(0)))),
((int)(((byte)(0)))));
            this.button14.Location = new System.Drawing.Point(198, 256);
            this.button14.Name = "button14";
            this.button14.Size = new System.Drawing.Size(83, 45);
            this.button14.TabIndex = 17;
            this.button14.Text = "Ok";
            this.button14.Visible = false;
            this.button14.Click += new
System.EventHandler(this.button14_Click);
            //
            // view6
            //
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Inherit;
            this.BackColor = System.Drawing.Color.White;
```

```
            this.ControlBorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.Controls.Add(this.label1);
            this.Controls.Add(this.button14);
            this.Controls.Add(this.button13);
            this.Controls.Add(this.textBox2);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.button12);
            this.Controls.Add(this.button11);
            this.Controls.Add(this.button10);
            this.Controls.Add(this.button9);
            this.Controls.Add(this.button8);
            this.Controls.Add(this.button7);
            this.Controls.Add(this.button6);
            this.Controls.Add(this.button5);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.ForeColor = System.Drawing.Color.White;
            this.Size = new System.Drawing.Size(640, 390);
            this.Text = "Insert Password";
            this.Controls.SetChildIndex(this.button1, 0);
            this.Controls.SetChildIndex(this.button2, 0);
            this.Controls.SetChildIndex(this.button3, 0);
            this.Controls.SetChildIndex(this.button4, 0);
            this.Controls.SetChildIndex(this.button5, 0);
            this.Controls.SetChildIndex(this.button6, 0);
            this.Controls.SetChildIndex(this.button7, 0);
            this.Controls.SetChildIndex(this.button8, 0);
            this.Controls.SetChildIndex(this.button9, 0);
            this.Controls.SetChildIndex(this.button10, 0);
            this.Controls.SetChildIndex(this.button11, 0);
            this.Controls.SetChildIndex(this.button12, 0);
            this.Controls.SetChildIndex(this.textBox1, 0);
            this.Controls.SetChildIndex(this.textBox2, 0);
            this.Controls.SetChildIndex(this.button13, 0);
            this.Controls.SetChildIndex(this.button14, 0);
            this.Controls.SetChildIndex(this.label1, 0);
            this.ResumeLayout(false);

        }

        #endregion

        // Escreve os correspondetes número e acrescenta um "*"
        // mostrada ao utilizador

        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text += 1;
            textBox2.Text += "*";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            textBox1.Text += 2;
            textBox2.Text += "*";
        }

        private void button3_Click(object sender, EventArgs e)
        {
            textBox1.Text += 3;
```

```
        textBox2.Text += "*";
    }

    private void button4_Click(object sender, EventArgs e)
    {
        textBox1.Text += 4;
        textBox2.Text += "*";
    }

    private void button5_Click(object sender, EventArgs e)
    {
        textBox1.Text += 5;
        textBox2.Text += "*";
    }

    private void button6_Click(object sender, EventArgs e)
    {
        textBox1.Text += 6;
        textBox2.Text += "*";
    }

    private void button7_Click(object sender, EventArgs e)
    {
        textBox1.Text += 7;
        textBox2.Text += "*";
    }

    private void button8_Click(object sender, EventArgs e)
    {
        textBox1.Text += 8;
        textBox2.Text += "*";
    }

    private void button9_Click(object sender, EventArgs e)
    {
        textBox1.Text += 9;
        textBox2.Text += "*";
    }

    // Limpa a caixa de texto para que o utilizador possa
    // reescrever a password em caso e engano
    private void button11_Click(object sender, EventArgs e)
    {
        textBox1.Text = "";
        textBox2.Text = "";
    }

    private void button10_Click(object sender, EventArgs e)
    {
        string path = "C:/FOOTWEAR/Applications/Password/password.txt";

        // Vai ler o ficheiro que contem a password
        // para que possa comparada

        StreamReader list = new StreamReader(path);

        string password = list.ReadLine();
        list.Close();

        // Comparar password com a escrita pelo utilizador

        if(password == textBox1.Text)
        {
```

```
                //Mensagem de texto se estiver correcta

                string message = "Password Accepted";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_2), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
            }
            else
            {
                //Mensagem de texto se estiver errada

                string message = "Password Wrong, try again";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_3), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
            }
        }

        // Password dada como correcta, vai abrir o menu load/erase

        private void OnServerMessageClosed_2(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
             this._view5 = new view5();
             this._view5.Closed += new EventHandler(view5_Closed);
             this._view5.ShowMe(this);
             textBox1.Text = "";
             textBox2.Text = "";
        }


        private void OnServerMessageClosed_3(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            this.Dispose();
        }

        void view5_Closed(object sender, EventArgs e)
        {
            this._view5.Closed -= new EventHandler(view5_Closed);
            this._view5.Dispose();
            this._view5 = null;
        }

        private void button13_Click(object sender, EventArgs e)
        {
                button14.Visible = true;
                button10.Visible = false;
                button13.Enabled = false;
                string message = "Insert current password";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_4), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);

        }
```

```
        private void OnServerMessageClosed_4(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
        }

        private void button14_Click(object sender, EventArgs e)
        {
            if (x == 2)
            {
                StreamWriter list = new
StreamWriter("C:/FOOTWEAR/Applications/Password/password.txt");
                list.Close();

                StreamWriter list_writer =
File.AppendText("C:/FOOTWEAR/Applications/Password/password.txt");
                list_writer.WriteLine(textBox1.Text);
                list_writer.Close();

                string message = "Password Changed";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_3), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
            }

            if (x == 1)
            {
                string path =
"C:/FOOTWEAR/Applications/Password/password.txt";
                //vai carregar para a variavel tr o conteudo do ficheiro
lista.txt

                StreamReader list = new StreamReader(path);

                string password = list.ReadLine();
                list.Close();


                if (password == textBox1.Text)
                {
                    textBox1.Text = "";
                    textBox2.Text = "";
                    x = 2;
                    string message = "Password Accepted, insert new
password";
                    string caption = "ATTENTION";

                    GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_4), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);

                }
                else
                {
                    string message = "Password Wrong, try again";
                    string caption = "ATTENTION";
                    GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_3), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
                }
```

```
            }
        }

        private void button12_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

    }

}
```

3.2.Submenu *Load/Erase Shoes*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ABB.Robotics.Tps.Windows.Forms;
using System.IO;

namespace TpsViewFootWearFinishing
{
    public class view5 : TpsForm
    {

        private view2 _view2;
        private ABB.Robotics.Tps.Windows.Forms.ListView listView1;
        private ABB.Robotics.Tps.Windows.Forms.Button button1;
        private ABB.Robotics.Tps.Windows.Forms.Button button2;
        private ABB.Robotics.Tps.Windows.Forms.Button button3;
        private TpsLabel tpsLabel7;
        private AlphaPad alphaPad1;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;


        public view5()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Clean up any resources used by this class.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                try
                {
```

```
                    if (disposing)
                    {
                        //ToDo: Call the Dispose method of all FP SDK
instances that may otherwise cause memory leak

                        if (components != null)
                        {
                            components.Dispose();
                        }
                    }
                }
                finally
                {
                    base.Dispose(disposing);
                }
            }
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(view5));
            this.listView1 = new ABB.Robotics.Tps.Windows.Forms.ListView();
            this.button1 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button2 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button3 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.tpsLabel7 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.alphaPad1 = new ABB.Robotics.Tps.Windows.Forms.AlphaPad();
            this.SuspendLayout();
            //
            // listView1
            //
            this.listView1.BackColor = System.Drawing.Color.White;
            this.listView1.BorderSize = 1;
            this.listView1.CheckBoxes = true;
            this.listView1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.listView1.ForeColor = System.Drawing.Color.Black;
            this.listView1.HeaderStyle =
System.Windows.Forms.ColumnHeaderStyle.Clickable;
            this.listView1.ItemHeigth = -1;
            this.listView1.LargeImageList = null;
            this.listView1.ListViewItemSorter = null;
            this.listView1.Location = new System.Drawing.Point(14, 66);
            this.listView1.MultiSelect = false;
            this.listView1.Scrollable = true;
            this.listView1.SelectedIndex = -1;
            this.listView1.SelectionEnabledOverScrollbuttons = false;
            this.listView1.ShowNumberOfItems = false;
            this.listView1.ShowSelection = false;
            this.listView1.Size = new System.Drawing.Size(230, 170);
            this.listView1.SmallImageList = null;
            this.listView1.Sorting =
ABB.Robotics.Tps.Windows.Forms.SortOrder.Ascending;
            this.listView1.SubItemsImageList = null;
            this.listView1.TabIndex = 7;
            this.listView1.View = System.Windows.Forms.View.LargeIcon;
```

```
            this.listView1.Zoomable = false;
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.White;
            this.button1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button1.BackgroundImage")));
            this.button1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button1.Location = new System.Drawing.Point(14, 282);
            this.button1.Size = new System.Drawing.Size(112, 95);
            this.button1.TabIndex = 8;
            this.button1.Text = "";
            this.button1.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.White;
            this.button2.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button2.BackgroundImage")));
            this.button2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button2.Location = new System.Drawing.Point(132, 282);
            this.button2.Size = new System.Drawing.Size(112, 95);
            this.button2.TabIndex = 9;
            this.button2.Text = "";
            this.button2.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.BackColor = System.Drawing.Color.White;
            this.button3.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button3.BackgroundImage")));
            this.button3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button3.Location = new System.Drawing.Point(524, 323);
            this.button3.Size = new System.Drawing.Size(112, 64);
            this.button3.TabIndex = 10;
            this.button3.Text = "";
            this.button3.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button3.Click += new
System.EventHandler(this.button3_Click);
            //
            // tpsLabel7
            //
            this.tpsLabel7.BackColor = System.Drawing.Color.Transparent;
            this.tpsLabel7.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel7.Location = new System.Drawing.Point(14, 31);
            this.tpsLabel7.Size = new System.Drawing.Size(147, 29);
            this.tpsLabel7.TabIndex = 11;
            this.tpsLabel7.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel7.Title = "Shoe\'s Type:";
            //
            // alphaPad1
```

```
            //
            this.alphaPad1.BackColor = System.Drawing.Color.White;
            this.alphaPad1.ControlBorderStyle =
System.Windows.Forms.BorderStyle.None;
            this.alphaPad1.EnableStringSelectorMenuItem = true;
            this.alphaPad1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.alphaPad1.FormAppearance = false;
            this.alphaPad1.Icon = null;
            this.alphaPad1.SelectedText = "";
            this.alphaPad1.Size = new System.Drawing.Size(639, 390);
            this.alphaPad1.TabIndex = 0;
            this.alphaPad1.Closed += new
System.EventHandler(this.alphaPad1_Closed_1);
            //
            // view5
            //
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Inherit;
            this.BackColor = System.Drawing.Color.Transparent;
            this.ControlBorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.Controls.Add(this.tpsLabel7);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.listView1);
            this.Text = "Load/Erase Programs";
            this.Load += new System.EventHandler(this.view5_Load);
            this.Controls.SetChildIndex(this.listView1, 0);
            this.Controls.SetChildIndex(this.button1, 0);
            this.Controls.SetChildIndex(this.button2, 0);
            this.Controls.SetChildIndex(this.button3, 0);
            this.Controls.SetChildIndex(this.tpsLabel7, 0);
            this.ResumeLayout(false);

        }

        #endregion

        private void button3_Click(object sender, EventArgs e)
        {
            //Mandar abaixo o alphanumérico para libertar memória!

            alphaPad1.Dispose();

            //Fechar a janela e ir para trás, menu password, libertando a
memória

            //do controlador

            this.Dispose();
        }

        private void view5_Load(object sender, EventArgs e)
        {
            //Vai carregar a lista de sapatos para a listview presente

            string path = "C:/FOOTWEAR/Shoes/Shoes.txt";

            StreamReader list = new StreamReader(path);

            /*ciclo, que enquanto nao chegar ao final do ficheiro
```

```
            vai criar novos items com o nome da existente em cada linha do
ficheiro
            e depois adiciona-los a' listbox */

            while (!list.EndOfStream)
            {
                ABB.Robotics.Tps.Windows.Forms.ListViewItem item = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem(list.ReadLine());
                listView1.Items.Add(item);
            }
            // fecha a leitura
            list.Close();
        }

        //Botão Eliminar sapato da lista

        private void button2_Click(object sender, EventArgs e)
        {
            if (listView1.SelectedIndex != -1 &&
listView1.SelectedItem.Checked == true)
            {
                //Questão de confirmação sobre certeza!

                string message = "Are you sure to delete this item?";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_2), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);
            }
            else
            {
                //Questão a avisar que não selecionou nada para apagar

                string message = "Nothing selected";
                string caption = "ATTENTION";
                GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_3), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
            }
        }
        private void OnServerMessageClosed_2(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            // se o resultado for YES, caso contrário não faz nada
            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {

                // index da selecção (VE=valor escolhido)

                int ve;
                ve = listView1.SelectedItem.Index;

                StreamWriter list = new
StreamWriter("C:/FOOTWEAR/Shoes/Shoes.txt");
                list.Close();

                int x;
                int y;

                x = listView1.Items.Count - 1;
```

```
                y = 0;

                //Escrever no ficheiro, ignorando o valor escolhido
                //De modo a poder elimina-lo da lista sem que fique um
espaço
                //em branco

                while (y <= x)
                {
                    if (y != ve)
                    {
                        StreamWriter list_writer =
File.AppendText("C:/FOOTWEAR/Shoes/Shoes.txt");
                        list_writer.WriteLine(listView1.Items[y].Text);
                        list_writer.Close();
                    }
                    y = y + 1;
                }
                //Apagar a corrente lista

                listView1.Items.Clear();

                //Actualizar para a nova lista sem o valor escolhido

                StreamReader list_1 = new
StreamReader("C:/FOOTWEAR/Shoes/Shoes.txt");

                /*ciclo, que enquanto nao chegar ao final do ficheiro
                vai criar novos items com o nome da existente em cada linha
do ficheiro
                e depois adiciona-los a' listbox */

                while (!list_1.EndOfStream)
                {
                    ABB.Robotics.Tps.Windows.Forms.ListViewItem item = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem(list_1.ReadLine());
                    listView1.Items.Add(item);

                }
                // fecha a leitura

                list_1.Close();
            }
        }

        private void OnServerMessageClosed_3(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
        }

        private void button1_Click(object sender, EventArgs e)
        {
            //Abrir teclado alphanumérico para inserir nome;

            alphaPad1.ShowMe(this);

        }

        private void alphaPad1_Closed_1(object sender, EventArgs e)
        {
            //Se o resultado for OK na chave alphanúmerica
            //Caso contrário, não faz nada
```

```
            if (this.alphaPad1.DialogResult ==
System.Windows.Forms.DialogResult.OK)
            {
                //Se o valor inserido não for nulo

                if (alphaPad1.SelectedText != string.Empty)
                {
                    //Ir para novamente para a janela de adicionar/remover

                    string texto;
                    texto = null;

                    //Guardar valor escrito
                    texto = alphaPad1.SelectedText;


                    //Vai aceder à base de dados da lista de sapato e vai
escrever
                    //uma linha de modo a adicionar o sapato que se
adicionou

                    StreamWriter list_write =
File.AppendText("C:/FOOTWEAR/Shoes/Shoes.txt");


                    list_write.WriteLine(texto);

                    ABB.Robotics.Tps.Windows.Forms.ListViewItem item = new
ABB.Robotics.Tps.Windows.Forms.ListViewItem(texto);
                    listView1.Items.Add(item);


                    // fecha a leitura

                    list_write.Close();


                    this._view2 = new view2();
                }

                //Caso a escolha tenha sido nula, uma mensagem de aviso é
enviada
                //ao utilizador

                else
                {
                    string message = "No text has been written";
                    string caption = "ATTENTION";
                    GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_3), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.OK);
                }
            }
        }
    }
}
```

4. **Código do submenu *DeBug***

```
using System;
using ABB.Robotics.Controllers.RapidDomain;
using ABB.Robotics;
using ABB.Robotics.Controllers;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ABB.Robotics.Tps.Windows.Forms;
using ABB.Robotics.Controllers.MotionDomain;
using System.IO;

namespace TpsViewFootWearFinishing
{
    public class view3 : TpsForm
    {
        Controller c = new Controller();
        Task tRob1 = null;
        private ABB.Robotics.Tps.Windows.Forms.Button button1;
        private ABB.Robotics.Tps.Windows.Forms.Button button2;
        private ABB.Robotics.Tps.Windows.Forms.Button button3;
        private ABB.Robotics.Tps.Windows.Forms.Button button4;
        private TpsLabel tpsLabel1;
        private TpsLabel tpsLabel2;
        private TpsLabel tpsLabel3;
        public float xx, yy, zz, pos1, pos2, pos3, pos4;
        public double q1, q2, q3, q4;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;


        public view3()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Clean up any resources used by this class.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                try
                {
                    if (disposing)
                    {
                        //ToDo: Call the Dispose method of all FP SDK
instances that may otherwise cause memory leak

                        if (components != null)
                        {
                            components.Dispose();
                        }
                    }
```

```
                }
                finally
                {
                    base.Dispose(disposing);
                }
            }
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(view3));
            this.button1 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button2 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button3 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button4 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.tpsLabel1 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel2 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.tpsLabel3 = new ABB.Robotics.Tps.Windows.Forms.TpsLabel();
            this.SuspendLayout();
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.White;
            this.button1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button1.BackgroundImage")));
            this.button1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button1.Location = new System.Drawing.Point(460, 104);
            this.button1.Size = new System.Drawing.Size(146, 138);
            this.button1.TabIndex = 8;
            this.button1.Text = "";
            this.button1.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // button2
            //
            this.button2.BackColor = System.Drawing.Color.White;
            this.button2.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button2.BackgroundImage")));
            this.button2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button2.Location = new System.Drawing.Point(241, 104);
            this.button2.Size = new System.Drawing.Size(146, 138);
            this.button2.TabIndex = 9;
            this.button2.Text = "";
            this.button2.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button2.Click += new
System.EventHandler(this.button2_Click);
            //
            // button3
            //
            this.button3.BackColor = System.Drawing.Color.White;
```

```
            this.button3.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button3.BackgroundImage")));
            this.button3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button3.Location = new System.Drawing.Point(18, 104);
            this.button3.Size = new System.Drawing.Size(146, 138);
            this.button3.TabIndex = 10;
            this.button3.Text = "";
            this.button3.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button3.Click += new
System.EventHandler(this.button3_Click);
            //
            // button4
            //
            this.button4.BackColor = System.Drawing.Color.White;
            this.button4.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button4.BackgroundImage")));
            this.button4.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button4.Location = new System.Drawing.Point(520, 328);
            this.button4.Size = new System.Drawing.Size(120, 62);
            this.button4.TabIndex = 11;
            this.button4.Text = "";
            this.button4.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button4.Click += new
System.EventHandler(this.button4_Click_1);
            //
            // tpsLabel1
            //
            this.tpsLabel1.BackColor = System.Drawing.Color.White;
            this.tpsLabel1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel1.Location = new System.Drawing.Point(28, 248);
            this.tpsLabel1.Size = new System.Drawing.Size(136, 29);
            this.tpsLabel1.TabIndex = 12;
            this.tpsLabel1.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel1.Title = "Stop Program";
            //
            // tpsLabel2
            //
            this.tpsLabel2.BackColor = System.Drawing.Color.White;
            this.tpsLabel2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel2.Location = new System.Drawing.Point(208, 248);
            this.tpsLabel2.Size = new System.Drawing.Size(221, 29);
            this.tpsLabel2.TabIndex = 13;
            this.tpsLabel2.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
            this.tpsLabel2.Title = "Deactivate Force Control";
            //
            // tpsLabel3
            //
            this.tpsLabel3.BackColor = System.Drawing.Color.White;
            this.tpsLabel3.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tpsLabel3.Location = new System.Drawing.Point(493, 248);
            this.tpsLabel3.Size = new System.Drawing.Size(91, 29);
            this.tpsLabel3.TabIndex = 14;
            this.tpsLabel3.TextAlignment =
System.Drawing.ContentAlignment.TopLeft;
```

```
            this.tpsLabel3.Title = "Go Home";
            //
            // view3
            //
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Inherit;
            this.BackColor = System.Drawing.Color.White;
            this.Controls.Add(this.tpsLabel3);
            this.Controls.Add(this.tpsLabel2);
            this.Controls.Add(this.tpsLabel1);
            this.Controls.Add(this.button4);
            this.Controls.Add(this.button3);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.button1);
            this.Size = new System.Drawing.Size(640, 390);
            this.Text = "DeBug";
            this.Controls.SetChildIndex(this.button1, 0);
            this.Controls.SetChildIndex(this.button2, 0);
            this.Controls.SetChildIndex(this.button3, 0);
            this.Controls.SetChildIndex(this.button4, 0);
            this.Controls.SetChildIndex(this.tpsLabel1, 0);
            this.Controls.SetChildIndex(this.tpsLabel2, 0);
            this.Controls.SetChildIndex(this.tpsLabel3, 0);
            this.ResumeLayout(false);

        }

        #endregion b

        //Click de botão Stop !

        private void button3_Click(object sender, EventArgs e)
        {
            //Questão sobre a certeza de iniciar, uma vez que não se
            //trata necessariamente de paragem de emergência

            string message = "Are you sure to start this performance?";
            string caption = "ATTENTION";
            GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_3), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);

        }
        private void OnServerMessageClosed_3(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            //Caso seja sim...

            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {
                //Paragem do Robô !
                Rapid rapid = c.Rapid;
                tRob1 = c.Rapid.GetTask("T_ROB1");
                rapid.Stop();

            }
        }

        // Click de botão GoHome !

        private void button1_Click(object sender, EventArgs e)
        {
```

```
            string message = "Are you sure to start this performance?";
            string caption = "ATTENTION";
            GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_1), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);


            //implementar programa de recurso ao uso de Yes ou No
        }
        private void OnServerMessageClosed_1(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            // Caso seja Yes
            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {
                //Ler posição do TCP do Robô

                Controller c = new Controller();
                MotionSystem motion = c.MotionSystem;
                MechanicalUnit mech = motion.GetActiveMechanicalUnit();
                RobTarget robP;
                robP = mech.GetPosition(CoordinateSystemType.Tool);

                StreamWriter str = new
StreamWriter("C:/FOOTWEAR/Applications/GoHome/Module1.mod");

                //Ler posição do TCP;

                xx = robP.Trans.X;
                yy = robP.Trans.Y - 60; //Tirar 60 mm à coordenada Y
relativamente ao TCP
                zz = robP.Trans.Z;

                // quatérnios

                q1 = robP.Rot.Q1;
                q2 = robP.Rot.Q2;
                q3 = robP.Rot.Q3;
                q4 = robP.Rot.Q4;

                // Ler configuração

                pos1 = robP.Robconf.Cf1;
                pos2 = robP.Robconf.Cf4;
                pos3 = robP.Robconf.Cf6;
                pos4 = robP.Robconf.Cfx;

                //Após ler posição, escrever programa Rapid
                //que, desativa o force control, recua 60 mm da posição
onde está,
                //e avança até à posição após recuo.

                str.WriteLine("MODULE Module1");
                str.WriteLine("\t" + "CONST robtarget offset:=[[" +
xx.ToString().Replace(",", ".") + "," + yy.ToString().Replace(",", ".") +
"," + zz.ToString().Replace(",", ".") + "]" + "[" +
q1.ToString().Replace(",", ".") + "," + q2.ToString().Replace(",", ".") +
"," + q3.ToString().Replace(",", ".") + "," + q4.ToString().Replace(",",
".") + "]" + "," + "[" + pos1.ToString() + "," + pos2.ToString() + "," +
pos3.ToString() + "," + pos4.ToString() + "],[9E9,9E9,9E9,9E9,9E9,9E9]];");
```

```
                str.WriteLine("\t" + "CONST robtarget
GoHome:=[[0,0,0,0,30,0],[9E9,9E9,9E9,9E9,9E9,9E9]];");
                str.Write("PROC Path_10()");
                str.WriteLine();
                str.WriteLine("\t"  + "MoveL
offset,v100,z100,tool0\\WObj:=wobj0;");
                str.WriteLine("\t" + "MoveAbsJ
GoHome,v100,z100,tool0\\WObj:=wobj0;");
                str.WriteLine("ENDPROC");
                str.WriteLine("PROC main()");
                str.WriteLine("\t" + "FCDeact;");
                str.WriteLine("\t" + "Path_10;");
                str.WriteLine("ENDPROC");
                str.WriteLine();
                str.WriteLine();
                str.WriteLine("ENDMODULE");
                str.Close();


                Rapid rapid = c.Rapid;
                tRob1 = c.Rapid.GetTask("T_ROB1");
                string path = "C:/FOOTWEAR/Applications/GoHome/GoHome.pgf";
                tRob1.LoadProgramFromFile(path, RapidLoadMode.Replace);

                //Executar o Programa Rapid

                rapid.ResetProgramPointer();
                rapid.Start();
            }
        }

        // Click de botão Desactivate Force Control

        private void button2_Click(object sender, EventArgs e)
        {
            //Questão sobre confirmação de desactivação do FC

            string message = "Are you sure to start this performance?";
            string caption = "ATTENTION";
            GTPUMessageBox.Show(this, new
MessageBoxEventHandler(OnServerMessageClosed_2), message, caption,
System.Windows.Forms.MessageBoxIcon.Question,
System.Windows.Forms.MessageBoxButtons.YesNo);

        }

        private void OnServerMessageClosed_2(object sender,
ABB.Robotics.Tps.Windows.Forms.MessageBoxEventArgs e)
        {
            //Caso Seja Yes

            if (e.DialogResult == System.Windows.Forms.DialogResult.Yes)
            {
                //Carregar programa em RAPID para desactivar FC

                Rapid rapid = c.Rapid;
                tRob1 = c.Rapid.GetTask("T_ROB1");
                string path =
"C:/FOOTWEAR/Applications/DeactivateFC/FCDeact.pgf";
                tRob1.LoadProgramFromFile(path, RapidLoadMode.Replace);

                //Executar o Programa Rapid
                rapid.ResetProgramPointer();
```

```
                rapid.Start();

            }
        }

        private void button4_Click_1(object sender, EventArgs e)
        {
            this.Dispose();
        }
       }

      }
```

5. Código do submenu *Help*

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using ABB.Robotics.Tps.Windows.Forms;

namespace TpsViewFootWearFinishing
{
    public class view4 : TpsForm
    {
        private ABB.Robotics.Tps.Windows.Forms.TabControl tabControl1;
        private ABB.Robotics.Tps.Windows.Forms.TabPage tabPage2;
        private ABB.Robotics.Tps.Windows.Forms.TabPage tabPage1;
        private PictureBox pictureBox2;
        private PictureBox pictureBox1;
        private ABB.Robotics.Tps.Windows.Forms.Button button12;
        private ABB.Robotics.Tps.Windows.Forms.Button button1;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;


        public view4()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Clean up any resources used by this class.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                try
                {
```

```
                    if (disposing)
                    {
                        //ToDo: Call the Dispose method of all FP SDK
instances that may otherwise cause memory leak

                        if (components != null)
                        {
                            components.Dispose();
                        }
                    }
                }
                finally
                {
                    base.Dispose(disposing);
                }
            }
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(view4));
            this.tabControl1 = new
ABB.Robotics.Tps.Windows.Forms.TabControl();
            this.tabPage2 = new ABB.Robotics.Tps.Windows.Forms.TabPage();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.tabPage1 = new ABB.Robotics.Tps.Windows.Forms.TabPage();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.button12 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.button1 = new ABB.Robotics.Tps.Windows.Forms.Button();
            this.tabControl1.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.SuspendLayout();
            //
            // tabControl1
            //
            this.tabControl1.BackColor = System.Drawing.Color.White;
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tabControl1.ImageList = null;
            this.tabControl1.Location = new System.Drawing.Point(0, 23);
            this.tabControl1.SelectedIndex = 1;
            this.tabControl1.Size = new System.Drawing.Size(640, 367);
            this.tabControl1.TabIndex = 2;
            this.tabControl1.TabPages.Add(this.tabPage1);
            this.tabControl1.TabPages.Add(this.tabPage2);
            //
            // tabPage2
            //
            this.tabPage2.BackColor = System.Drawing.Color.White;
            this.tabPage2.Controls.Add(this.button1);
            this.tabPage2.Controls.Add(this.pictureBox2);
            this.tabPage2.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
```

```
            this.tabPage2.ImageIndex = -1;
            this.tabPage2.Location = new System.Drawing.Point(0, 40);
            this.tabPage2.Size = new System.Drawing.Size(640, 327);
            this.tabPage2.TabIndex = 2;
            this.tabPage2.Text = "DeBug";
            //
            // pictureBox2
            //
            this.pictureBox2.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox2.Image")));
            this.pictureBox2.Location = new System.Drawing.Point(0, 0);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(640, 327);
            //
            // tabPage1
            //
            this.tabPage1.BackColor = System.Drawing.Color.White;
            this.tabPage1.Controls.Add(this.button12);
            this.tabPage1.Controls.Add(this.pictureBox1);
            this.tabPage1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.tabPage1.ImageIndex = -1;
            this.tabPage1.Location = new System.Drawing.Point(0, 40);
            this.tabPage1.Size = new System.Drawing.Size(640, 327);
            this.tabPage1.TabIndex = 1;
            this.tabPage1.Text = "Process Shoes";
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(0, 0);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(640, 324);
            //
            // button12
            //
            this.button12.BackColor = System.Drawing.Color.White;
            this.button12.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button12.BackgroundImage")));
            this.button12.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button12.Location = new System.Drawing.Point(3, 262);
            this.button12.Size = new System.Drawing.Size(120, 62);
            this.button12.TabIndex = 14;
            this.button12.Text = "";
            this.button12.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button12.Click += new
System.EventHandler(this.button12_Click_1);
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.White;
            this.button1.BackgroundImage =
((System.Drawing.Image)(resources.GetObject("button1.BackgroundImage")));
            this.button1.FontName = new
ABB.Robotics.Tps.Windows.Forms.DesignerFont("Font12b");
            this.button1.Location = new System.Drawing.Point(3, 262);
            this.button1.Size = new System.Drawing.Size(120, 62);
            this.button1.TabIndex = 15;
            this.button1.Text = "";
```

```
            this.button1.TextAlign =
ABB.Robotics.Tps.Windows.Forms.ContentAlignmentABB.MiddleCenter;
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // view4
            //
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Inherit;
            this.BackColor = System.Drawing.Color.White;
            this.ControlBorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.Controls.Add(this.tabControl1);
            this.Size = new System.Drawing.Size(640, 390);
            this.Text = "Help";
            this.Controls.SetChildIndex(this.tabControl1, 0);
            this.tabControl1.ResumeLayout(false);
            this.tabPage2.ResumeLayout(false);
            this.tabPage1.ResumeLayout(false);
            this.ResumeLayout(false);

        }

        #endregion

        private void button12_Click(object sender, EventArgs e)
        {

        }

        private void button12_Click_1(object sender, EventArgs e)
        {
            this.Dispose();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Dispose();
        }

    }
}
```